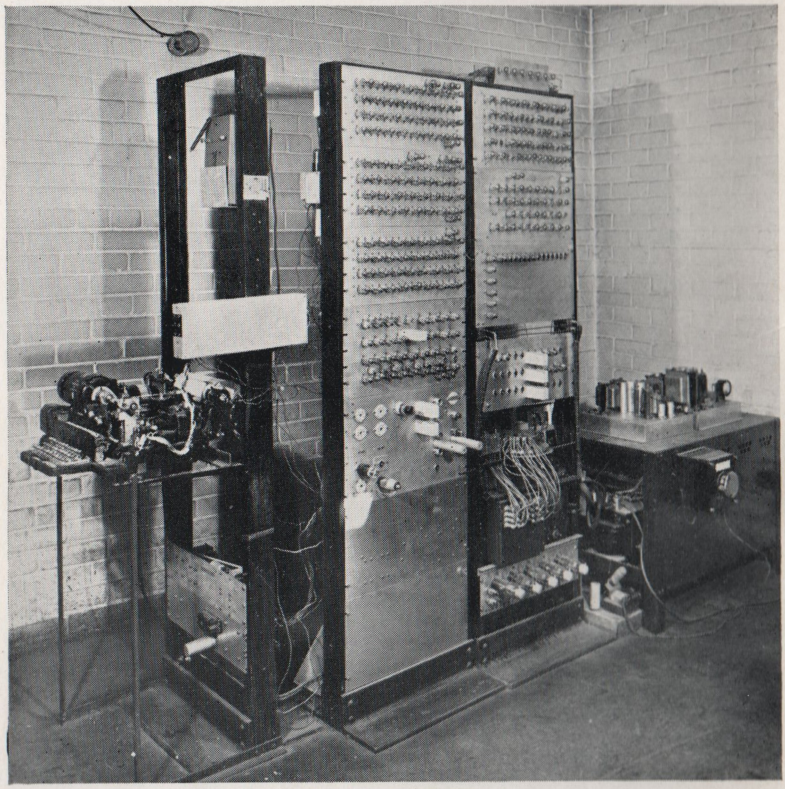
AUTOMATIC DIGITAL CALCULATORS

BOOTH & BOOTH



Frontispiece

The A.P.E.(R.)C.

AUTOMATIC DIGITAL CALCULATORS

By

ANDREW D. BOOTH,
D.Sc., Ph.D., F.Inst.P.

and

KATHLEEN H. V. BOOTH,
B.Sc., Ph.D.

LONDON
BUTTERWORTHS SCIENTIFIC PUBLICATIONS
1953

2087

BUTTERWORTHS PUBLICATIONS LTD., 88 KINGSWAY,
LONDON, W.C.2

AFRICA:	BUTTERWORTH & CO. (AFRICA) Ltd. DURBAN: Goodricke's Buildings, Masonic Grove
AUSTRALIA:	BUTTERWORTH & CO. (AUSTRALIA) Ltd. SYDNEY: 8 O'Connell Street, MELBOURNE: 430 Bourke Street BRISBANE: 240 Queen Street
CANADA:	BUTTERWORTH & CO. (CANADA) Ltd. TORONTO: 367 Danforth Avenue
NEW ZEALAND:	BUTTERWORTH & CO. (AUSTRALIA) Ltd. WELLINGTON: 49/51 Ballance Street, AUCKLAND: 35 High Street

U.S.A. Edition published by
ACADEMIC PRESS INC., PUBLISHERS
125 EAST 23RD STREET
NEW YORK 10, NEW YORK

Set in Monotype Baskerville type
Made and printed in Great Britain by William Clowes and Sons, Limited, London and Beccles

2087

PREFACE

THE decade 1940–1950 was one of intense activity in many fields of science. Under the spur of war it proved necessary to perform, rapidly and frequently, complex calculations which were beyond the scope of available human computers. It thus happened that the 19th century dream of Charles Babbage became the practical reality of the 20th century, and the automatic digital calculators of today form an important factor in the planning of most new departures in the sciences, and in engineering.

The present book is intended to serve as a guide to the theory of these machines, and particularly to introduce new workers in the field to the notions, language and literature of the subject. Not unnaturally, the techniques described in most detail are those in use in our own laboratory; but since, in the last seven years, most of the current suggestions have received investigation there, the account should not prove too biased.

The terminology of the subject is not yet stabilized, and it is hoped that readers with strong feelings will not take exception to our use of such an anthropomorphic term as 'memory' which, in any case, must be made familiar to any potential reader of the existing literature. It is perhaps unfortunate, however, that the advanced state of production of the book made it impossible to adopt the spelling 'program' instead of programme, since, at the 1953 Conference on High Speed Digital Computers held at the National Physical Laboratory, this revision found general acceptance. Although, in defence of the old variant, it may be pointed out that the past tense, and personal noun, then require no modification. This should appeal to advocates of simplified English.

The authors wish to acknowledge their indebtedness to John von Neumann and his staff, at the Institute for Advanced Study, Princeton, for the stimulating period spent as the guests of their Project. Thanks are also due to the British Rubber Producers' Research Association, Imperial Chemical Industries, Ltd., and to the Rockefeller Foundation, for their generous support which made the construction of A.P.E.(X.)C. type machines possible.

In conclusion, it gives us particular pleasure to thank Mr. John Wilson, now Director of Research at the British Rayon Research Association, for his patience, encouragement and very practical support over the past years.

A.D.B.
K.H.V.B.

The Wharf,
Fenny Compton,
May 1953.

CONTENTS

		PAGE
Frontispiece		ii
PREFACE ...		v
1.	INTRODUCTION ...	1
2.	THE MECHANICAL ERA	6
3.	THE ADVENT OF ELECTRONIC TECHNIQUES	13
4.	THE OVERALL DESIGN OF A COMPUTING SYSTEM	22
5.	THE CONTROL ...	27
6.	THE ARITHMETIC UNIT	35
7.	MISCELLANEOUS OPERATIONS	57
8.	INPUT—OUTPUT	61
9.	GATES ...	75
10.	SINGLE DIGIT STORAGE	90
11.	MISCELLANEOUS COMPONENTS	103
12.	STORAGE DEVICES	113
13.	DEFINITION OF A CODE AND DISCUSSION OF ITS FORM AND CONTENTS	136
14.	THE TECHNIQUE OF CODING	151
15.	THE USE OF SUBROUTINES IN CODING	158
16.	PROGRAMME DESIGN	180
17.	SOME APPLICATIONS OF COMPUTING MACHINES ...	197
BIBLIOGRAPHY		217
NAME INDEX		227
SUBJECT INDEX ...		228

1

INTRODUCTION

BEFORE coming to the main subject-matter of this book it seems appropriate to give a short introductory account of the general ideas current today in the field of computing machinery. This is particularly necessary in view of the wealth of technical jargon which has grown up during the past decade, and which makes it difficult for those working in the subject to remember when their remarks involve a background which is not a part of the general knowledge of the non specialist.

Many calculations which are familiar in everyday life involve the repetition of certain cycles of operation a great number of times. For example, in evaluating the deduction of Income Tax from the wages of workers in a factory or office, the same small set of rules apply to each worker, the only difference being the amount of his or her income and the various allowances depending on such things as number of children. If a machine were set up to compute Income Tax, it would only be necessary to insert the particular data mentioned above and the operation of the machine would then be automatic. This idea of a computer performing a fixed cycle of operations a number of times on different sets of input data is known as 'iteration' and is fundamental in the operation of a modern calculator.

In the scheme mentioned above, a machine was postulated which performed a fixed set of operations upon input data. The mechanism which causes the correct sequence of operations to occur is known as the 'control', and those parts of the machine which perform the actual additions, subtractions, multiplications, *etc.*, are known as the 'arithmetic unit'.

In the simple scheme pictured above the machine was assumed to have been constructed to perform one set of operations, to solve one particular problem. Such a machine is nowadays known as a 'special purpose computer'. Computers have now been built which are capable of being set up to solve any problem amenable to treatment by the rules of arithmetic and these are called 'general purpose computers'.

It is quite possible to perform calculations upon numbers by representing them by various physical quantities and then, using some mechanical or electrical device, to generate the required answer. A familiar example of this type of calculator is the slide rule; here the numbers are converted into lengths on a logarithmic scale, and products and quotients generated by the superposition of these quantities followed by their evaluation on a suitably calibrated scale. A device which works by such physical analogy is usually known as an 'analogue computer'.

Many and complex analogue computers exist and are of great beauty and elegance; they will not, however, form the subject matter of this book which is concerned with 'digital computers'. As their name implies these machines accept numbers in the form of digits and operate upon them by a process of counting. An elementary example of this type of computer with which the reader will be familiar is the cash register.

The great advantage of digital computers, over the analogue variety, lies in their potential accuracy. Thus it is extremely difficult to construct an analogue computer whose accuracy is much higher than one part in a hundred, whilst a digital machine can be extended to any accuracy required by the simple process of adding extra digits in the form of counting wheels and actuating mechanism.

The question may be asked, of what value is an accuracy of one part in ten million? In everyday life such precision is seldom, if ever, required; in mathematical calculation, however, it frequently happens that a desired small number is calculated as the difference of two large numbers so that great precision may be needed in the latter to make the former significant. (The question as to whether there exists a better mathematical method of approach to all such problems is here left open since, even if such methods exist, they may not always be the simplest from a computational point of view.)

Again, if a calculation on numbers of, say, two decimal digits involves a multiplication, in order to preserve two decimal digit precision the resulting product has to be 'rounded off.' This means that the answer may have an error of the order of one-half in the last digit.

It may be shown by a statistical process, that the sum of n numbers, each having a probable error $\frac{1}{2}$, is likely to have an error of $\sqrt{n}/2$. Thus the sum of 40,000, two decimal digit numbers having the above probable errors, will have a probable error of about 100. If the numbers had arisen in a problem where they alternated in sign and the resulting sum was small, it is evident from the above argument that the round-off error on the sum might exceed the

actual sum itself. Modern high speed digital computers perform arithmetic sequences of at least the magnitude mentioned above, and it is therefore important to maintain a high precision in the numbers used.

A modern digital computer may operate at a speed which makes it impossible for a human operator either to supply it with data at an adequate rate, or to write down the numbers as fast as the machine can compute them. This necessitates the provision of 'terminal organs' or 'input/output' devices. Such devices may take the form of paper tape on which numbers can be represented by punched holes, and of electric typewriters or punches operated by the machine itself. The use of a tape, read by the machine, makes it possible to employ a number of human operators to keep the machine occupied. Moreover there exists a vast range of problems, both mathematical and physical, which, while involving an enormous amount of calculation, require little input and generate only a small output. Such problems occur in the determination of molecular structure by x-ray methods, in the solution of many problems in nuclear physics, in the design of aircraft for supersonic flight, and in innumerable purely mathematical problems of the type: for what values of n is $2^n - 1$ prime?

Because of the expense and difficulty of building machines with large numbers of electrical or electronic components, most modern general purpose computers are constructed so that they can perform only a limited number of arithmetic operations. Thus addition and subtraction are universal, multiplication facilities exist on most machines but a divider is frequently absent and such things as square root units are almost unknown.

Any problem which it is required to solve must be split up into a sequence of these basic operations before the machine can be used; the process of arranging the problem in this way is called 'programming' or 'coding' and the set of basic operations which the machine can perform is called the 'code' or order set.

The idea of repeating or iterating a fixed sequence of operations a number of times has already been mentioned; suppose now that a machine is required to go through such a sequence just n times and then to stop. If the set of operations had to be inserted into the machine n times a great amount of equipment, time and space would be wasted. To overcome this the machine is so arranged that each time it comes to the end of the sequence it examines a certain number; if this number is positive or zero, say, it returns to the start of the sequence as before; if the number is negative, however, it stops or goes into a new sequence. Thus, suppose that as a part of the

original sequence the number $(n-1)$ has unity subtracted from it after each iteration and that the result is examined, for the first $(n-1)$ cycles the result will be positive so that the sequence is repeated; after the nth cycle, however, the result will be negative and the new sequence will be initiated. The number $n-1$ is called an 'iteration index', and the instruction which examines the value of $n-1$ and directs the machine to one of two courses of subsequent action is called a 'conditional transfer' or 'branching order'.

The conditional transfer is perhaps the most important order in the repertoire of a modern computer. It can be shown that even the most advanced logical processes can be programmed by using it and it can be made to direct the machine into alternative courses of action in a manner initially unknown to the human operator. As an elementary example of this may be mentioned the problem of extracting the square root of a number generated during the operation of the machine. This will be considered in detail in Chapter 15, but it suffices to say here that the process of extraction applied to a number a is an iterative one which generates at each iteration, m, an approximation x_m to $\sqrt{a}$. The problem is that the number of iterations required to produce a result of given accuracy depends on the magnitude of a and is initially unknown to the operator. The solution is simple. At the end of each iterative cycle the machine forms $-(x_{m-1}-x_m)^2$. This number is negative so long as $x_{m-1} \neq x_m$ to the precision required; at the end of the iteration however, the difference is zero so that an application of the conditional transfer procedure will take the machine out of the square root extraction cycle and on to the next part of the problem.

To conclude this introduction it is appropriate to make a brief remark upon the manner of representation of numbers in a computing machine.

In normal arithmetic a negative number would be represented by, say, $-0{\cdot}642$. This presentation is not particularly suitable for the numbers used in a computing machine, instead the notation $9{\cdot}358$ $(=10-0{\cdot}642)$ would be used, in which the negative number has been subtracted from a power of 10 greater than the maximum capacity of the machine. This process of representation is known as 'complementary', and if 10^p is the greatest power of 10 within the capacity of the machine, negative numbers are said to be represented as complements mod 10^{p+1}. It is easily seen that, within the capacity of the machine, $a-b=a+(\text{comp. } b \bmod 10^{p+1})$.

In many digital computing machines, especially those using electronic circuits, it turns out that the normal decimal scale is not that best suited to engineering practice. Instead a scale involving

only the digits 0 and 1 is most convenient; this is known to mathematicians as the 'binary scale' or scale of 2. Leibnitz suggested long ago that this scale should replace the decimal scale for school teaching in view of its simple tables.

$$\begin{array}{ll} 0+0=0 & 0\times 0=0 \\ 1+0=1 & 0\times 1=0 \\ 1+1=10 & 1\times 1=1 \end{array}$$

There are, however, good reasons why this suggestion was never adopted, which the reader will appreciate if he tries to add a column of more than about three figures in this scale. In the binary scale the decimal digits appear as:

$$\begin{array}{l} 0=0 \\ 1=1 \\ 2=10 \\ 3=11 \\ 4=100 \\ 5=101 \\ 6=110 \\ 7=111 \\ 8=1000 \\ 9=1001 \end{array}$$

and any decimal number can be represented in binary form. It may be questioned whether difficulties will arise from the use of such a scale in a machine; this is not so since the machine can be programmed to accept information in decimal scale *via* a keyboard or punched tape: the numbers so inserted are converted by the machine into binary form for ease of calculation and storage. When the main calculation is complete the machine re-converts the numbers to decimal form for output to the human operator. This process is described in detail in Chapter 14 where the coding of problems receives attention.

Finally it should be noted that complement notation is again used for the representation of negative numbers in binary scale, complements now being taken mod 2^{p+1} where 2^p is the most significant digit which can be held in the machine registers.

2

THE MECHANICAL ERA

ARITHMETICAL MACHINES

THE earliest reference to the possibility of constructing a calculating machine capable of performing the operations of ordinary arithmetic appears to be due to Pascal who, in 1642, constructed a machine which was capable of performing addition and subtraction. He applied this machine, with considerable success, to tax collection in France.

The general idea of mechanical calculation strongly appealed to Leibnitz who, in 1694, constructed an extended version of Pascal's machine which had the faculty of multiplication and division.

Great interest attaches to the development of these first crude machines from the stage of being scientific curiosities to their present compact and reliable form, a development in which the English genius for mechanism found full expression in the early calculators of Morland and of Stanhope. It is perhaps worthy of mention, in passing, that the stylus operated adding machine of Morland is still sold in a largely unmodified form, as a low price instrument.

Any considerable commercial development of machines for performing the common operations of arithmetic had necessarily to await the availability of sound production engineering techniques suitable for the precision mass production of the numerous small gears and other mechanism required in their construction. Thus, whilst Thomas de Colmar, in 1820, put on the market a commercial machine, this was not particularly reliable.

The first arithmetical machines of the type now known were produced by Ohdner in 1891, although the variable toothed wheel which bears his name was, in fact, described by Roth in 1841 and used in an actual machine by Baldwin in 1875.

It is an interesting commentary upon scientific progress that one of the most compact and beautiful calculators recently available (1952) has reverted to the original Leibnitz stepped wheel as its main element.

Semi-automatic Machinery

The machines so far described depend upon the presence of a human operator to supply them with data, to cause their operation either manually or by the intermediary of an electric motor, and to record and re-order their results for the next stage of the calculation. Pascal, in his *Pensées*, says of calculating machines, with particular reference to his own, '*La machine arithmetique fait des effets qui approchent plus de la pensée que tout ce que font les animaux; mais elle ne fait rien qui puisse faire dire qu'elle a de la volonté comme les animaux*' and upon this doubtful evidence, has been described as the father of modern large scale computing machinery. The quotation seems altogether too vague for such a credit especially in view of the second clause.

Leibnitz has also the reputation of being a parent of self-operating machines; it has not proved possible to find a first-hand reference to support this and it seems unlikely that Leibnitz made any proposals of sufficient detail to substantiate his claims to priority.

The first documented suggestion for a partially automatic machine seems to be that of Müller who in 1786 proposed the construction of a device for the generation of functions from algebraic formulae involving their differences. This machine was apparently never made and it appears doubtful if detailed designs were prepared.

Charles Babbage designed a difference engine in 1822 and applied to the Government of the day for funds to construct it. This machine was to compute automatically, any function the first five of whose differences were given, and furthermore to produce a printed record of the calculations. It was based on a small model which 'produced figures at the rate of 44 a minute and performed with rapidity and precision all those calculations for which it was designed', and was commenced in 1823. By 1829 a large part of the machine had been constructed but difficulties with the engineering staff had led to an unsatisfactory state of the accounts and to trouble with the Treasury. When about £17,000 had been expended the constructional engineer, Mr. Clement, made such exorbitant demands for compensation that support was withdrawn. Work finally ceased in 1833 and, despite voluminous correspondence, a concrete statement of policy was not forthcoming from the Government until 1842 when the project was finally abandoned. The whole of the machine so far completed was generously offered as a gift to Babbage; this, however, was refused.

The completed parts of Babbage's difference engine have, since 1842, had a chequered career. In 1843 they were deposited in

King's College, London; after many years of neglect they were transferred to the Science Museum in South Kensington. Before leaving the subject of the difference engine it is worth remarking that the main cause of delay in the actual manufacture lay in the absence of precision machinery as it is now known. Babbage set out first to construct such devices and it is in no small measure due to his exertions in this direction that Great Britain achieved the pre-eminent position in engineering which it occupied by the end of the nineteenth century.

Although Babbage did not complete a difference engine, one constructed according to his design was built in Sweden by Scheutz and gave good and reliable service, being later moved to the Dudley Observatory, Albany, U.S.A.

It is perhaps not out of place to mention here the subsequent history of partially automatic calculating machines. Babbage himself suggested, in connection with his analytical engine *vide infra*, the use of punched cards, similar to those used in the Jacquard loom, to store data and instructions for a calculator. This plan was adopted in 1889 by Hollerith in a machine developed for data sorting in connection with the American census. From this device have developed the complex sorting, calculating and tabulating machinery used at the present time for business administration. Two main types of punched card machine exist; in the first, or *Powers Samas* machine, cards are read and calculations performed by exclusively mechanical means in a manner which is the spiritual descendent of that suggested by Babbage. The second, or *Hollerith* type of machine, uses the electromagnetic relay and other similar devices as computing elements, thus effecting a simplicity and flexibility impossible in the days of Babbage.

As far as the generation of functions from differences is concerned, the National Accounting machine with its 6 registers allows the performance and printing of exactly those calculations for which Babbage's difference engine was designed.

THE UNIVERSAL MACHINES

By 1833 Babbage had come to consider the possibility of constructing a calculating machine which would be capable of performing any calculation specified to it by the operator, and not merely calculations of a particular type as was the case with the difference engine.

This machine Babbage called the 'Analytical Engine.' In its design Babbage's genius came into full flower; it is not an exaggeration to say that all of the logical devices rediscovered for the building

of the latest types of electronic machine were initially thought of by Babbage.

In the first place the machine was to consist of two main parts, the 'mill' and the 'store.' The former, as its name suggests, was to be capable of working on, or with numbers in accord with the common operations of arithmetic. The latter unit was to consist of 1,000 columns of 50 counting wheels each to be available, under instruction, to the mill and other parts of the machine.

The wisdom of Babbage can be seen when it is remarked that an adequate storage capacity is today considered to be just about this size.

In order to insert data and instructions into the machine Babbage proposed to use a modification of the punched control cards long in use for the weaving of complex designs in the Jacquard loom. In the latter device cards are linked together permanently in sequence which would, of course, prevent rearrangement. Whilst it is true that Babbage does not make explicit mention of the fact that in his machine the cards are not to be connected, it seems clear that such was his intention since he mentions that stacks of cards are to be inserted into the machine. Thus the punched card operated business machine is clearly foreshadowed in Babbage's initial work.

Next, concerning the mill itself, Babbage realized clearly that in order to have a reasonable speed of arithmetic operation it was necessary that the accumulating units of a machine should have the faculty of recognizing the presence of a chain carry and of jumping from the first to the last stage. Thus, to take a simple example, if 1 be added to 899,999,999, the machine is to recognize the long string of 9's and consequent carries and to liquidate the latter simultaneously whilst adding unity to the 8. Mechanism for this process was designed by Babbage and incorporated in the analytical engine.

So far as output is concerned, Babbage projected mechanism whereby the machine would set up a block of type representing the results of its calculations ready for printing. In addition it would check the correctness of the type so set and in this respect it is markedly superior to any machine constructed up to the present.

Finally Babbage proposed that his machine should be capable of determining its course of action between several alternatives, in a way not previously known to the human operator. This is the so-called 'conditional transfer' of a modern computer and its inclusion in the analytical engine is yet another indication of the genius of its inventor.

When Babbage died, in 1871, the analytical engine was still incomplete. His son, Major-General H. P. Babbage, constructed a part of the mill in the years between 1880 and 1910 and at a demonstration this calculated and printed a table of multiples of π to 20 decimal places. The 1910 model and some fragments of Babbage's original mechanism are to be seen in the Science Museum at South Kensington.

The logical descendent of Babbage's analytical engine was not to be constructed for nearly 70 years after his death. In 1937 Howard Aiken of Harvard University began work on an 'Automatic Sequence Controlled Calculator' now generally referred to as 'A.S.C.C.' or Mk. 1. This machine, built with the assistance of the International Business Machine Co., was completed in 1944. In some ways it is a much more modest device than that envisaged by Babbage. It contains 60 constant registers and 72 adding registers compared with the 1,000 planned by Babbage. Units are provided for multiplication and division and also electromechanical tables for $\log_{10} x$, 10^x and $\sin x$. A special device is incorporated for interpolating between tabular values of the above functions, and the main sequencing of the machine is performed by perforated paper tapes. The normal length of the numbers used in the machine is 24 decimal digits.

Following the construction of the A.S.C.C. the Harvard group commenced in 1945 the construction of a calculator, usually called Mk. 2, in which all arithmetic and transfer operations are performed by means of electromagnetic relays. This machine, completed in 1947, contains 13,000 relays of a special design. The use of the relay, as a substitute for mechanism, makes possible a considerable increase in speed, thus the times of operation in Mk. 2 are:

Transfer to or from store	33 msec
Addition or subtraction	200 ,,
Multiplication	700 ,,

The storage capacity of the machine is 100 numbers each of 10 decimal digits and a sign.

Elaborate input and output facilities are provided and output can be typed at the rate of 15×25 characters per minute, alternatively it may be punched on tape, for future use in the machine, at the rate of 40×25 per minute. Finally the machine may be split into two complete units and operated independently.

Other pioneer work in relay computing was carried out by the Bell laboratories starting in 1938. The first of a series of machines was completed in 1940 and since that time three other models have

been constructed. These computers were designed for fire control applications, but in 1944 a general purpose machine was built. Two models of this computer have been constructed; 9,000 telephone relays are used and a large number of teletype units. Decimal numbers coded in bi-quinary scale are used in the machines (in this scale digits 0–4 and 5 are available and each decimal digit appears as the closure of only *two* of the available relays thus allowing a simple check to be made), the numbers are of 7 decimal digit precision together with a sign, and the machine was the first to use floating point notation. Forty-four storage registers are available in relay form and arithmetic facilities for the more usual operations are provided. Times of operation are:

Addition or subtraction	0·3 sec
Multiplication	1·0 ,,
Division	2·2 ,,
Square root	4·3 ,,

A particular advantage of this machine lies in the extensive checking facilities built into it which enable long problems to be run unattended.

Commercially available relay calculators have been marketed by the *International Business Machine Co.*, under the name *Pluggable Sequence Relay Calculator*. This machine follows closely the standard design for I.B.M. equipment, it has 36 storage registers of 6 decimal capacity. Arithmetic operations and times are:

Addition or subtraction	25 msec
Multiplication	900 ,,
Division	1200 ,,

A particular advantage of this machine lies in its use of punched cards for input/output which provides great flexibility and also a speed well matched to the internal speed of the machine.

Other large scale relay or electromechanical computers are the *B.A.R.K.* in Stockholm, the *Züse* computer and The Imperial College Computing Engine.

In 1947 a study was made by the present authors of the possibility of constructing general purpose computers of much greater simplicity than those either available or projected. This work culminated in the design and construction of the Automatic Relay Computer (A.R.C.). This had, in its initial form, storage for 256 numbers on a magnetic drum and in consequence should really find its place in the next chapter; however, the magnetic drum store has now been replaced by an extremely compact electromechanical

store for 50 numbers together with a pluggable sequence unit for 300 instructions so that it can be properly described here.

The machine operates in true binary scale with numbers of 21 binary digit precision (5 decimal). Operating times are:

Transfer, addition or subtraction	20 msec
Multiplication or division	1000 ,,

Input and output are on standard teletype equipment with an auxiliary keyboard for test operation. The notable feature of the machine lies in the fact that it uses only 800 single-contact high speed relays; a number which could be further reduced by using multi-contact relays. The relatively high speed of the machine is attained by parallel operation and the accumulator has chain carry facilities of the type envisaged by Babbage. A particular difference between this machine and others currently available lies in its use of binary scale throughout, conversion being programmed to and from the decimal output and input devices.

A second machine the A.R.R.A. originally based on the A.R.C. design has been constructed in the Mathematical Centre, Amsterdam.

3

THE ADVENT OF ELECTRONIC TECHNIQUES

WHEN the technical developments of the radio industry had made possible the large scale production of wireless valves, the way was open for the application of these devices to the construction of computing machinery. It is surprising, in retrospect, that the step was so long in coming, since the characteristics required of valves in computing machine use are far less complex than those needed for wireless telephony.

The key electronic discovery (after that of the three electrode valve) was made as long ago as 1919, when Eccles and Jordan showed how a pair of triode valves could be connected to form a circuit having two stable states. At any time, from this date onward, a computer of modern type could have been constructed. In case the *cognoscenti* should argue that the problem of storage had not been solved, it should be pointed out that notwithstanding the absence of the cathode ray tube and the mercury delay line from the field of available devices, the Poulsen magnetic recorder had been in existence for nearly 25 years and the basic suggestion of magnetic data storage had been made by Oberlin Smith in 1888.

The computational problems which might form the spur to the development of an electronic computing machine had existed for nearly a century in the astronomical field, and were in active request by the Quantum Chemists of the 1930's. Despite the presence of all these favourable factors, it was not until the early 1940's that serious consideration was given to the engineering problem of constructing an electronic calculator and, in fact, it is not too much to say that the 1940's have been the era of electronics in their wider sense, and as distinct from wireless applications.

SPECIAL PURPOSE DIGITAL COMPUTATION

Under the stress of war with its need for extensive ballistic tables, the U.S. War Department sponsored the development of a special purpose electronic computer for these calculations. This machine,

called the 'Electronic Numerical Integrator and Calculator' (ENIAC), was designed by Eckert and Mauchley and constructed at the Moore School of Engineering, Philadelphia. After a public disclosure in 1946 the machine was moved to the Ballistic Research Laboratory, Aberdeen, Maryland, where it has been in operation since 1947.

Although primarily designed for projectile calculation, it was soon realized that ENIAC could be used for a wide variety of other calculations of a general character, and to this extent it is perhaps unfair to class the machine as special purpose.

Nevertheless, the basic design of ENIAC lacks many of those characteristics foreseen by Babbage, and its use in any particular problem requires a mathematical *tour de force* so that the classification seems fair.

ENIAC was constructed without regard for space and cost; it contains some 18,000 valves, consumes 100 kW of electric power and occupies a room over 100 feet in length. The machine is provided with a plugging system so that the various units can be connected together and sequenced to suit the particular problem to be solved. This interconnection process is very tedious but has been replaced, to some extent, by the use of a function table (*vide* Chapter 11) for sequencing in a way suggested by von Neumann.

Decimal scale is used throughout and 20 accumulators are provided, each capable of holding a 10 decimal number and its sign. Typical operating times of the machine are:

Addition and subtraction	200 μsec
Multiplication	2,300 ,,

and units are provided for division and extraction of the square root. It is also possible to store functional values required in the course of a calculation; these are set by hand on resistance matrix function tables which may, as mentioned above, also be used for easy insertion of the programme.

Input and output to ENIAC are by way of standard punched card equipment and, in view of the limited internal storage of the computer, form the chief limitation on its speed of operation.

In the years since its construction, ENIAC has been used for the solution of a very wide range of problems notable among which are the solution of partial differential equations, and the more recent calculation of the values of π and e to 2,000 decimal places. Despite its undoubted limitations and design complexity, ENIAC marks, perhaps, the most important single step in the history of applied

mathematics, and it provided the encouragement and backing for the more versatile machines described in the next section.

GENERAL PURPOSE ELECTRONIC DIGITAL CALCULATORS

Charles Babbage clearly foresaw the need for a large storage capacity to be available in a general purpose calculator. When ENIAC was under construction, von Neumann and his co-workers devoted considerable attention to the problem of the optimum form of a calculating machine. As a result of this it was shown that a calculator should have a storage capacity of the order of 1,000 words and that, given this, both operating instructions and pure numbers could be stored in the same unit. The way in which this is achieved is simple; each instruction contains (or implies) the position in the store not only of the numbers upon which operations are to be performed, but also of the next instruction. Thus if the machine is started in a position known to contain an instruction, it can select, in the correct sequence, those words which are really instructions (as distinct from pure numbers) from the mass of numerical data held in the store. The parenthetical 'implies' given above refers to an alternative mode of operation in which instructions are stored in consecutive positions so that a simple counting process can be used to locate them when the starting position is known. Means must be provided for jumping out of sequence either unconditionally or in the light of some criterion, but both of the above schemes are readily extended to meet this need.

The storage capacity of 1,000 words mentioned above, posed by itself a considerable problem. Each word must consist of a number of digits and it was shown that an appropriate length was between six and twelve decimals. Furthermore, to make adequate use of the high intrinsic speed of the electronic valve, it was desirable that the storage device used should be able to emit data, after request, in a time of the order of a few millionths of a second, and should be capable of absorbing it at the same rate. Even at the present time this *desideratum* has not been fully met, but in the mid 1940's the only device which seemed capable of operation at the required speed was the mercury acoustic delay line. The delay line will be considered in detail in Chapter 12, it suffices to say here, that if a number can be represented by a set of impulses, spaced out in time, it can, in principle, be stored by sending these impulses into some medium which will delay them (without alteration) and re-emit them at a later time. If the head and tail of such a device are connected together, data, once inserted, will circulate

indefinitely; but it should be noted that, on the average, a waiting period of the order of one-half of the total delay time is needed before a word can be read from or written in an arbitrary position.

The original proposals of von Neumann led to the design of an electronic general purpose computer—the Electronic Discrete Variable Automatic Computer or EDVAC, which uses the sonic delay properties of mercury as a store and employs only about 3,500 valves. This machine, projected in 1941 at the Moore School, has only recently been completed; it suggested, however, two other projects which produced more rapid results. The first of these was the Electronic Delayed Storage Automatic Computer or EDSAC, which was completed at the Mathematical Laboratory of the University of Cambridge in 1949. This machine, constructed under the direction of M. V. Wilkes, has a storage capacity, in mercury delay lines, of 512 words each having 34 binary digits; it employs some 3,000 valves. Facilities are provided for operating on words of only 17 binary digit precision and, in this mode, the addition/subtraction time is about 34 μsec. For full length numbers addition or subtraction take approximately 70 μsec and for multiplication about 8·5 msec are required.

Input to the EDSAC is *via* standard five hole teletype tape, photo-electrically read, and output is to a teletype tape punch or teleprinter. A particular advantage of the EDSAC lies in the carefully constructed library of subroutines, by means of which complex problems can be solved by assembling together methods of solution previously used on simpler calculations. It should be noted that the EDSAC was the first machine, operating on the principle of a large store containing both numbers and instructions, to be put into actual computing service.

The second project for a high speed electronic computing machine using delay line storage was initiated, in 1945, at the National Physical Laboratory, London, under the direction of Womersley, Turing and Colebrook. The Automatic Calculating Engine, or ACE, was completed in 1950, it has a storage capacity of 512 words of 32 binary digits. Addition and subtraction take 32 μsec and multiplication requires about one msec. Input and output are *via* standard punched card equipment. The ACE is particularly well designed, it employs less than 1,000 valves and is compact in structure, this simplicity has resulted in a reliability of operation which is probably unequalled at the present time (1952).

It should be stated that what has been described above as the ACE, is, in reality, only a pilot model for a much larger machine of the same name. Both the EDVAC and the EDSAC operate in

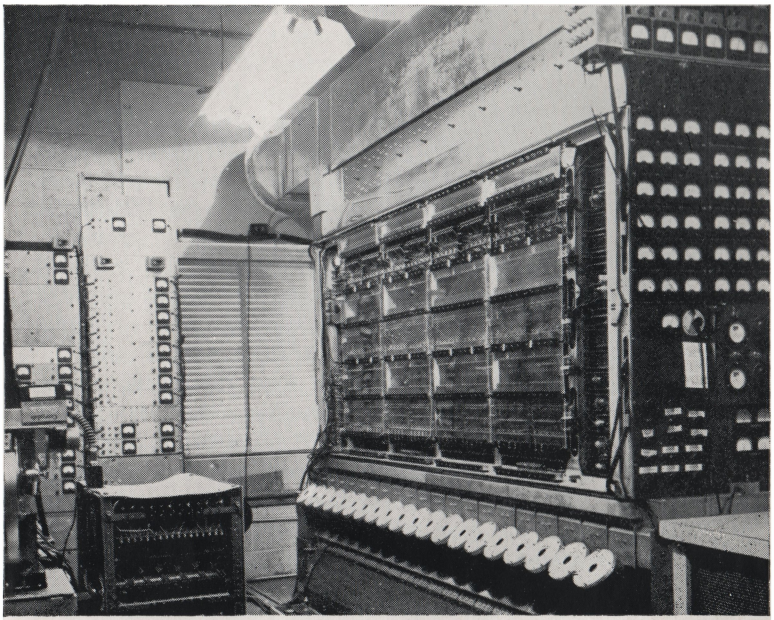


Plate 1. The Institute for Advanced Study Computer, Princeton

the mode in which instructions are placed in consecutive positions in the store; the ACE (pilot), however, adopts the so-called 'two address code' in which each instruction contains, not only the location of the number to be operated upon, but also that of the next instruction. This procedure allows the use of optimum programming techniques (see Chapter 16) to reduce the waiting time inherent in delay line storage devices.

Several other machines have been or are being constructed on the basis of the acoustic delay line store. Notable among these is the Bureau of Standards Eastern Automatic Computer, or SEAC, at Washington, which makes extensive use of semi-conducting devices as replacements for thermionic valves, and is designed to accept newer forms of storage device than the delay line when these become available. The characteristics of SEAC are roughly the same as those of ACE; it was first used on problems in 1951. Another machine of similar type, but working in the decimal scale, is the UNIVAC. This machine was constructed for the U.S. Bureau of Census by Eckert and Mauchley, builders of the ENIAC; it possesses what is probably the most comprehensive input and output system in existence, most of these functions being performed by magnetic tape. As an interesting aside it may be remarked that the UNIVAC was used to predict, from small sample analysis, the results of the 1952 American Presidential Election with, it may be stated, complete accuracy! Other machines of similar type are under construction but reliable information as to their characteristics is lacking; available references are given in the bibliography.

Following the initial projects for the construction of computers based upon the storage of data in serial form, in sonic delay lines, there was a period devoted to the consideration of alternative forms of storage device and of the optimum logical design for the machines themselves. On the logical side, Burks, Goldstine and von Neumann, at Princeton, examined the necessary components and best form of code for a machine. In a set of now classical reports starting in 1946, the foundations were laid for the design of most of the best computers available today. The Princeton group projected the construction of a machine working in parallel, binary mode and depending for its data storage upon a device known as the 'Selectron' for the high speed element, and upon magnetic wire for the input and output. Eventually the Selectron was abandoned in favour of a somewhat simpler—though slower—form of storage, and the input/output device replaced by the now usual punched paper tape. (This is to be again replaced by punched card equipment.) The Institute for Advanced Study computer has been in service since

March 1952, it operates on words of 40 binary digits used in parallel; internal storage is on a modified Williams cathode ray tube system and has 1,024 word capacity; addition and subtraction take approximately 10 μsec and multiplication requires 300 μsec. This machine is probably (taking account of the large word length) the fastest in operation today (1952); the punched paper tape input/output system is at present the bottleneck since nearly 20 min are required to fill or empty the high speed store from this medium.

Mention has been made of the Selectron and of cathode ray tube storage. These make use of certain secondary emission properties of dielectric surfaces when bombarded by an electron beam and a more detailed description must await Chapter 12; it is sufficient to say here, that these devices enable digital data to be stored in a form which does not require the long access time inherent in delay lines and, in consequence, make possible, in principle at least, the construction of much higher speed calculators than before.

Two main applications of cathode ray tube storage have been made. The first is at the Servomechanisms Laboratory of the Massachussetts Institute of Technology, where a very fast computer, the Whirlwind, has been built under the direction of J. Forrester. This machine uses a cathode ray tube store of a type devised by Forrester, which requires two sources of electrons in the tube; the Whirlwind has storage for over 1,000 words, each of 16 binary digits, it employs 6,000 valves and is capable of performing addition and subtraction in 5 μsec. Multiplication requires 40 μsec. Input and output are normally *via* punched paper tape and teletype, but an elegant device is provided which enables the machine to display its output, in graphical form, on the face of a large cathode ray tube where it may be inspected visually, or recorded photographically.

The second main type of cathode ray tube storage is that pioneered by F. C. Williams at the University of Manchester, England. In this system (see Chapter 12) standard cathode ray tubes are used with consequent advantages in cost and availability. The Williams computer, as constructed by *Ferranti Ltd.*, is considerably slower than the Whirlwind. Storage for 256 words of 40 binary digit precision is provided, addition and subtraction require 1·2 msec and for multiplication more than 3 msec are required. Three thousand valves, in addition to the cathode ray tubes, are used. Input is *via* punched paper tape read photo-electrically, and output is to a teletyper. An intermediate store is provided on a magnetic drum.

Other machines, using the cathode ray tube store in some form, are those at the Telecommunications Research Establishment

(TRE), Malvern, England, at the Institute for Numerical Analysis, California, U.S.A. (SWAC) and at the University of Illinois, U.S.A. (ORDVAC).

The third form of storage device makes use of the magnetic remanence of ferromagnetic materials. The first system of this type to be used in a computing machine was constructed by the present authors in 1947. This employed a nickel-plated cylinder which rotated at high speed and was arranged with reading and recording heads to enable binary data to be recorded in the form of magnetized elements on its surface. The first fully electronic machine constructed with storage on this plan was the prototype Simple Electronic Computer (SEC) at the Electronic Computation Laboratory of Birkbeck College, University of London. By very careful logical design this machine employed only 230 valves to give a storage of 256 words of 21 binary digit precision, addition and subtraction required 1·6 msec; in view of the experimental nature of the machine no multiplier was provided. A notable feature of SEC was the use of a two-address code, which enabled optimum coding techniques to be used to reduce the waiting time inherent in a cyclic storage device. Following the operation of SEC, several All Purpose Electronic Computers were constructed on the same lines (the letters in parentheses are used to specify the agency for whom the machine was constructed). So far, four machines have been built:

APE(X)C—For Birkbeck College, London.
APE(N)C—For the Board of Mathematical Machines, Oslo, Norway.
APE(H)C—For the British Tabulating Machine Company.
APE(R)C—For the British Rayon Research Association.

The first two machines have storage for 1,024 words of 32 binary digits and utilize about 420 valves. Addition and subtraction require about 600 μsec, and multiplication takes up to 20 msec. Input and output devices range from teletype equipment on APE(R)C and APE(N)C to punched-card card machines for APE(X)C and APE(H)C. It is hoped that machines of this type will soon be commercially available from the British Tabulating Machine Company.

Several other machines using magnetic drum storage have been constructed; notable among these are the Mk. 3 and Mk. 4 Calculators at the Computation Laboratory, Harvard, the Engineering Research Associates ERA 1101 Calculator, the University of

California CALDIC, and the Circle Computer at New York University.

It is not to be supposed that the brief survey of computers given above is complete, several machines are not included, either because no adequate reference can be found, or because they are known to be secret. New projects appear almost daily and their numbers will probably increase as the techniques become better established, the most up-to-date details available are given in the bibliography and to it the reader requiring more detailed information is referred.

Before proceeding to the next section of the book, it is perhaps worth while surveying what has been achieved and considering possible future trends. Two major problems are apparent in current computer technology; the first is that of finding a really efficient and cheap storage device, and the second is that of eliminating the electronic valve from the computer.

All existing data storage devices, the acoustic delay line, the cathode ray tube, and the magnetic drum, suffer from the extreme disadvantage that their output is only of the order of one millionth of their input power, this means that a large amplification is required to recall stored data and this, in turn, makes reliable operation difficult. Of the three devices mentioned above, only the magnetic drum is a true store, in the sense that it will retain its data without power input, and, in principle, any desired probability of correctness of output from the drum can be attained by reading out the word a number of times and comparing the results. This technique is not possible with the delay line or the cathode ray tube, but in any case is not a true solution to the problem since it makes the operation of the store undesirably slow.

Experiments are now in progress, at Birkbeck College, London, and in the Servomechanisms Laboratory, Massachussetts Institute of Technology, directed at using very small toroids of ferromagnetic material, arranged at the lattice points of a net of single conductors, as a storage device. This technique, first suggested by one of the authors in a lecture at Cambridge in 1947 under the name of the 'iron nail store,' seems very promising; access times of the order of a few microseconds are easily attained and, despite its use of macroscopic elements for storage, the device is compact and reasonably inexpensive. The output can be made as large as one-tenth of the input power so that, in this respect, the arrangement is much superior to any at present in use.

A further development, at the Servomechanisms Laboratory, and also at the Bell Telephone Laboratory, gives great promise for the future; in this use is made of ferro-electric materials of the barium

titanate type. Storage devices can be made from thin sheets of ceramic-like material upon which the input and output conductors are drawn with conducting paint; it is hoped that the available output may, with development, be made of the same order as the input.

The elimination of thermionic valves from computing machines is almost universally favoured by designers. Valves consume considerable energy in the form of heater dissipation, they are fragile, and their life is limited. Work at present in progress seems to indicate that semi-conducting devices may replace the valve. Already germanium crystal diodes find considerable use, in selected locations, and the transistor amplifier may soon attain a reliability which will make it a rival to the triode valve. Quite aside from these developments, it is already possible to eliminate the valve as a gating element and to replace it by a magnetic structure of equivalent speed but of effectively infinite life.

4

THE OVERALL DESIGN OF A COMPUTING SYSTEM

WHATEVER type of computing machine is projected, so long as it is to have the attribute of universality, it must necessarily have in its structure some component capable of performing at least the most elementary operation of arithmetic–subtraction. It is not necessary that any more complex operation of the type of multiplication be provided as an automatic function, although it appears that if the machine is to be reasonably fast and not to make excessive demands on the capacity of its storage elements, the provision of at least some additional arithmetic functions is desirable.

Broadly speaking, then, the computing system of an all-purpose computer can be likened to an ordinary desk calculating machine of any well-known type, and it is instructive to consider a few statistics in connection with the latter variety of device. In the first place, for automatic models, the time of an addition is of the order of 0·5 sec and that for multiplication about 10 sec. Now an analysis of the time schedules of a large number of fairly complex computations shows that for each operation of the machine about 50 sec are spent in referring to numbers and computational formulae on sheets of paper and in inscribing interim results on other sheets.

It therefore follows that, even if additions were relatively few compared with multiplications, the machine could not be working more than about one-sixth of the available computing time, and in general a much smaller duty cycle will prevail.

These observations make it clear that little or nothing is to be gained by increasing the speed of the arithmetic unit, as the computing part of the machine will be called, unless such an increase can be coupled with an increased speed of using the unit and of recording its data. Taking account of the fact that with relatively unsophisticated electronic techniques, addition times of 5–10 μsec and multiplication times of 100–1000 μsec are possible, it becomes evident that some electronic equivalent of the pencil and paper and also of the computing sequence are necessary. These units are

usually called the 'Memory' and the 'Control,' and in the next sections their characteristics will be briefly discussed.

THE MEMORY

This, as explained above, is to be a high speed equivalent of the pencil and paper used by a human computer. Clearly it must have at least two functions.

(1) Data must be inscribable in it ('writing').
(2) It must be capable of emitting inscribed data when required ('reading').

Furthermore, since many computations can be reduced to iterative processes, it is most desirable that it should be possible to erase data recorded in any given memory location and to replace it by new material.

With most of the storage devices suggested up to the present, a more or less long time elapses between the receipt of an order to 'read' or 'write' and its execution by the system. This delay is called the 'access time' of the memory, and for some typical units varies from 10 μsec to 10 msec.

Considering next the computing sequence itself, it is evident that, unless some particular sequence is to be built into the fabric of the machine, which is contrary to the postulate of universality made at the start of this chapter, some memory must be provided for the instructions or 'orders.' It would, of course, be possible to provide a separate memory for this purpose, but it will now be shown that such a procedure is illogical and results in wastage of components. To do this it is merely necessary to remark that most computations fall into one of two classes; in the first, a large number of operations are to be applied to a relatively few numerical quantities and the results of the calculations are to be few in number; in the second, few instructions suffice for the disposal of a large stock of data. As examples, may be mentioned the solution of some types of ordinary differential equation and the evaluation of the sums of squares and products of observed data, these problems being of the first and second types respectively. Whence, a machine would, to be of universal application, require to have two large memories, one of which would usually be only partially filled. The question thus naturally arises, can one unit be used for both functions? The answer is yes, and it is easy to see how to achieve this objective; one solution would to be affix to the groups of digits representing orders, some extra digit to distinguish them from pure numbers. This,

however, would be wasteful of memory capacity and is unnecessary since another more satisfactory procedure is available. In general, the orders required to perform a given calculation will follow one another in an orderly sequence, only rarely (relatively) will it be necessary to jump over portions of the table or to reiterate the procedure. This means that if orders are stored in sequence in the memory, the control can select them by a process of counting as soon as it knows the location of the start of the sequence. Similarly, it can jump to other positions containing orders if the last instruction in any sequence always makes a reference to the location of the start of the next. Thus, all that is necessary, is to have the machine, on being switched on, refer to memory location 1 and to arrange that an order is always recorded there. Two courses are then possible, either the control may contain a counter, which is advanced one position on receipt of a signal that an order has been executed, and whose contents then give the memory location of the next order (except of course when the control 'jumps' out of sequence, in which case the contents of the counter are exchanged bodily to record the new position), or each order contains a record of the memory location in which the next order is to be found. The latter procedure is very flexible and leads to numerous useful 'dodges' but has the disadvantage of wasting valuable memory space to record these extra locations.

All storage devices can be divided into two classes—serial and parallel. In the former, the digits of a stored number become available one at a time, usually (although not necessarily) starting from the least significant, in the latter all the digits of a number become available simultaneously.

The organization of the remainder of the machine, and especially of the arithmetic unit, depends to a very large extent upon the particular class to which the memory belongs. For example, when all digits of a number appear simultaneously it is clearly desirable and proper to operate with them as a group in performing an addition. Not so, however, for serial operation since in this case the digital pattern would first have to be transferred to some form of parallel storage device and this operation followed by another distinct operation commanding addition as a group; but it is clear that if the sum is formed one digit at a time as the number appears from the memory, the whole process will be complete as soon as the read out operation terminates, thus saving the extra operation mentioned above.

In practice, it is sometimes necessary to deviate from this strictly logical design principle since, for certain types of serial storage

device, the access time may be so great as to make multiplication a prohibitively slow operation if performed in a strictly serial manner. This subject will be taken up in greater detail in Chapter 6 when discussing the organization of the arithmetic unit.

THE CONTROL

Some idea of the nature of the control will have been gathered from the discussion of the mode of storage of orders in the memory, the constituent parts of this important organ will now be enumerated. Since the control must be capable of receiving the coded order from

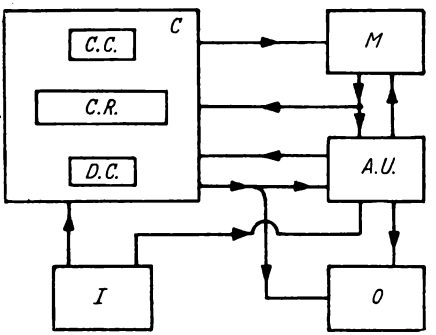


Figure 4.1. Block diagram of complete machine

the memory and storing it during the execution process, which will generally involve the separate use of the memory itself, it is evidently necessary to have at least one storage device in the control unit. This is called the 'control register.' The fact that the control may also contain a counter has already been mentioned. This must be interconnected with the control register in such a way that on receipt of an instruction of the form:

'Jump to the order located in memory position (x)' the contents of the counter can be modified so as to read (x). In practice, as will be seen in Chapter 5, it may be more convenient to have a complete adder instead of a counter; this will count in the ordinary manner if impulses are applied to its units position, but can be changed to read (x) by the simple process of adding $(x-c)$ into it, (c) being its contents before the addition takes place.

In order to store instructions in the memory, it is necessary that they be expressed in some coded numerical form. Thus certain digits out of the group comprising a complete order will specify

the memory location in which the relevant data is to be found, whilst others will comprise the instruction to the elements of the machine regarding the required operation on the data. This last group, consisting initially of digits, must be necessarily translated in electrical impulses to actuate the machine. This process is called 'decoding' and the control must contain a unit to perform this function, it is called the 'decoder.' Apart from switch or 'gate' elements, the foregoing are the principal components of the control so that *Figure 4.1* gives an outline of the structure of a typical machine. It is to be noted that the elements which direct the arithmetic unit in its special functions are not included in the control described here. They are assumed to be included in the arithmetic unit itself and are called the 'local control.' A description will be found in Chapter 6.

THE TERMINAL ORGANS

Finally it is obviously necessary to be able to insert data into the machine and also for the machine to communicate the results of its calculations to the operator. These functions are performed by two 'terminal organs' called respectively, the 'Input' (*I*) and the 'Output' (*O*).

5

THE CONTROL

IT HAS been seen in the preceding chapter that the control of an all-purpose computing machine may have at least three distinct functional units, register, counter and decoder. The next stage is to examine the manner in which these are effective in ordering the operations of the machine.

It is difficult to be completely general in describing the unit, since its structure and operation depend to some extent on the nature of the memory and of the arithmetic unit, but it is hoped that the outline given will be sufficient to make evident the mode of operation in any particular case.

THE ORDERS

Before entering upon any description of the control itself, it is necessary to have some idea of the types of order which require execution, and of the way in which these are represented in numerical coded form.

Briefly speaking, in binary code a typical order would appear as follows:

1001	1000	0101	1111	0110	1101
a	*b*	*c*	*d*	*e*	*f*

Groups *a* and *b* together give the location of the memory position from which it is desired to extract (or to which it is desired to send) data. Group *c* and the first digit of *d* give, in coded form, the operation which it is required to perform; and in a certain type of computer, *e* and *f* might contain the memory location of the position in which the next instruction is to be found, although, for this type of machine, no control counter would be required. Thus the code shown above would mean:

Add the absolute value of	*c* and *d*, i.e. 01 01, 1 (111)
The number contained in memory position 152	*a* and *b*, i.e. 1001, 1000 (=152)
Into the accumulator	(*c* and *d* again)

and possibly: then start to locate the next order which is to be found in memory position 109 (i.e. e and $f = 0110,\ 1101: = 109$).

In general the orders to be executed by the machine can be divided into six main types.

(1) Involving operations in the arithmetic unit only with no reference to a number stored in the memory. For example, shifts and transfers of numbers.

(2) Arithmetic operations involving transfer of data from the memory to the arithmetic unit. Such as addition, subtraction, multiplication and division.

(3) Transfers of numbers from the arithmetic unit to the memory.

(4) Transfers of the control, to new routines, located outside the main sequence, or transfers within that sequence. Such transfers not involving the use of any data not contained in the instruction itself.

(5) Conditional transfers of the control out of sequence of the type: if the number N (located in the arithmetic unit) is $\geqslant 0$ proceed in sequence with the orders but if $N < 0$ jump to the order located in memory location (x) and proceed in sequence from there.

(6) Orders involving the terminal organs (input/output).

It will be seen, later, that the control described in this chapter is capable of executing all instructions of these types.

THE DECODER

Since the operation to be performed is initially represented by the state of certain elements (electronic 'flip-flops' or electromagnetic relays) which themselves represent a number, it is evidently necessary to have some means of translating this information into action on the part of the machine.

For most machines projected up to the present, the total number of distinct operations, which can be performed, is less than 32. This means that, if each operation is given a code number, not more than 32 such code numbers will be required and, in binary scale, five places will be required.

Thus a device is required which has 5 inputs and 32 outputs and works in such a manner that for each combination of the inputs there exists a unique live member of the 32 outputs. Furthermore, since several operations will have various sub-operations in common, it is clearly desirable that mechanism exist for having some of

the 32 outputs operate the same sub-operations without the possibility of internal feedback in the system.

These functions are performed by the decoder, which is seen to consist of two parts.

(1) A 'many-one' function table.
(2) A 'one-many' function table.

This is shown schematically in *Figure 5.1*, and further details of this unit are given in Chapter 11.

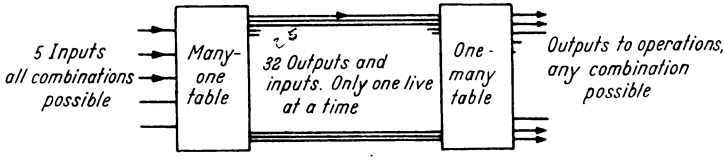

5 Inputs all combinations possible
Many-one table
32 Outputs and inputs. Only one live at a time
One-many table
Outputs to operations, any combination possible

Figure 5.1. Decoder

THE CONTROL COUNTER

Little need be said of this unit, it can take two forms according to the way in which the particular machine operates. In the first it is a straightforward counter which can be zeroed and reset to any required contents by simultaneous application of these to each of its stages; in the second form it consists of a complete parallel adder, that is, it can form and store the sum of the number which it already contains and any incident number.

It will be seen, in the section on coding, that the latter form of counter enables considerable simplification to be made in the use of pre-prepared subroutines.

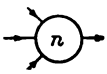
THE CONTROL REGISTER

This is a simple unit having as many stages as there are digits in the standard length word stored in the memory. It must be capable of receiving such words from the memory and of storing them for as long as required. It must also be capable of connection with the various operational units of the control itself (e.g. the decoder and the counter).

In order to make possible the drawing of simple schematics for the workings of the machine, it is convenient to have certain

'shorthand' notations for the various components. To describe the control the following prove convenient:

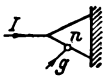
C.C.	= Control counter
D.C.	= Decoder
C.R.	= Control register
M.	= Memory
A.U.	= Arithmetic unit



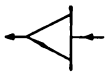
= A gate whereby an output is produced *only* if simultaneous inputs occur on each of the *n* inputs.

= The input to a functional unit having *n* channels.

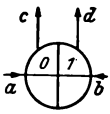
= The output from a functional unit having *n* channels.



= An *n* channel input (*I*) which is such that it affects the functional unit only on receipt of a gating pulse on the input *g*.



= A delay. Usually inserted to enable some operation to be completed prior to the commencement of the next.



= A binary element in its normal (0, 1) state. It has outputs on *c* and *d* which reverse on the element being triggered into an excited state by inputs on *a* or *b*.

= An auxiliary register which stores the location of the position from which the memory is required to read data.

THE MEMORY

Before describing the particular control circuit selected for discussion, it is necessary to say a few words about the memory itself, since the mode of operation of this unit influences the structure of the control.

Memories can be of two general types, parallel and serial. In

the former all the digits of a number become available simultaneously, whilst in the latter they arrive one at a time starting from the least significant. In both types there is in general a more or less great delay before the emission of the data begins.

For the purposes of the following description it will be assumed that the memory has the following characteristics.

(1) Capacity for 512 words each of 32 binary digits.
(2) Is of serial type.
(3) Emits data pulses continuously.
(4) Emits a group of 32 clock pulses coincident with the required data which can be used to 'gate' the output data pulses to the appropriate receptor.
(5) Emits an operation complete pulse after the data markers have all been transmitted.

The record operation is effected by using the 32 gating pulses, emitted at the correct word location, to cause the data stored in the arithmetic unit to be shifted digit by digit to the right, at the same time recording the value of the previous right-hand digit in the memory.

AN ACTUAL CONTROL SCHEMATIC

In *Figure 5.2* is shown the control schematic for the all-purpose electronic computing machine (A.P.E.(X.)C.) and it will now be explained how this operates to guide the machine in each type of order specified on p. 28.

First it will be assumed that a group of order digits is just appearing at the digit output of *M.* and that the binary element is set in the excited position (1, 0) so that the 2 gate g_1 is open and the train of 32 clock pulses passes through to *C.R.* These pulses cause the contents of *C.R.* to shift progressively to the right and, at each stage, one of the incident digits from *M.* is absorbed. When the whole 32 have appeared these will be stored in *C.R.* and the memory emits an operation complete pulse which restores the binary element to its normal state (0, 1). As this process of restoration to zero occurs, a pulse is emitted, *via* line *a* which gates the 9 memory location digits contained in *C.R.* to the memory location register *M.L.*; also, after a delay to enable the decoder to settle down on the contents of *C.R.*, a gating pulse is applied to the 16 functional outputs of *D.C.* and also to the gate g_3 which is concerned with absolute transfers of control. At this stage several

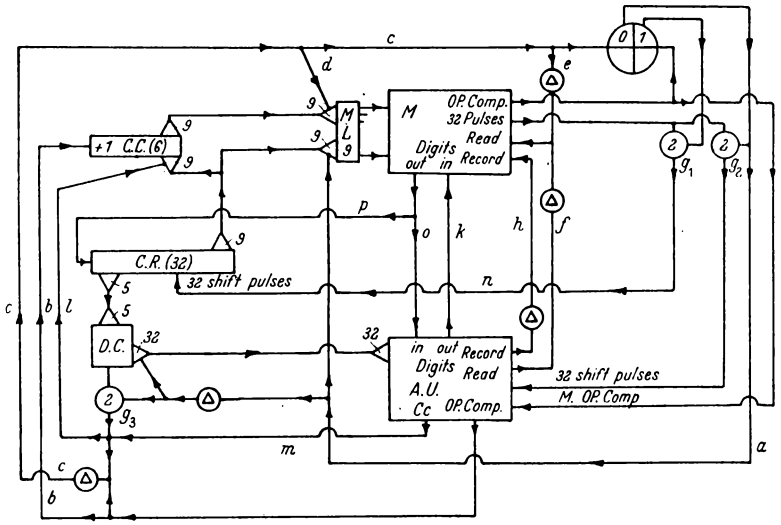


Figure 5.2. Schematic of one-address control

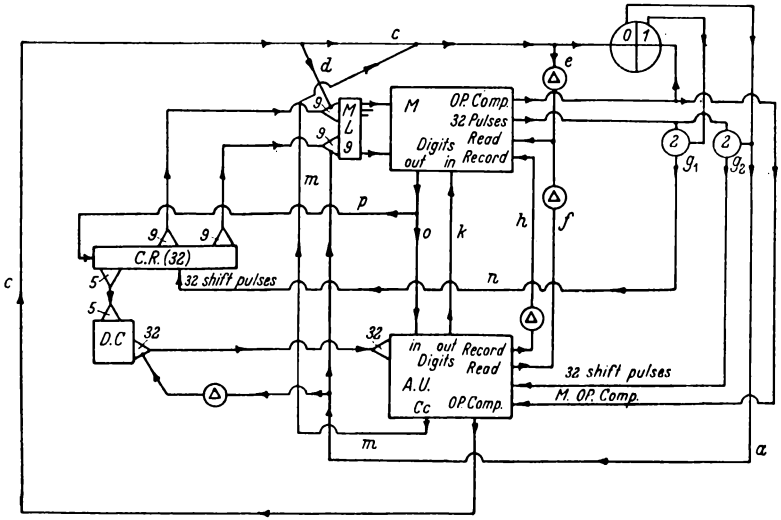


Figure 5.3. Schematic of two-address control

courses of action are possible and will be considered in the order specified on p. 28.

(1) Orders not involving the memory. Here the gate pulse applied to the 16 functional outputs of *D.C.* actuates the particular sequence line in *A.U.* *A.U.* performs its operation and then emits an operation complete pulse. This operation complete pulse advances the control counter one step *via* line *b* and then, after a delay $\triangle$ to allow *C.C.* to settle down, causes *M.L.* to absorb the position recorded by *C.C.* *via* lines *c* and *d*. At the same time the binary element is sent into its excited state (*via c*) and the memory is instructed to emit the order located in the position shown by *M.L.* (*via c* after a delay $\triangle$ to allow the binary element and the associated gate g_1 to function). When the memory has reached the appropriate position the clock digits are emitted and pass *via* g_1 to repeat the operations already specified.

(2) Arithmetic orders involving transfer from *M.* The gate pulse applied to the 16 functional outputs of *D.C.* acts as before and the arithmetic unit organizes itself to execute the required order. When it is ready it emits a 'read' pulse *via f* to the memory. The memory when ready emits 32 clock pulses at the given word, these pass, *via* g_2 (because the binary element is now in the unexcited (0, 1) state), to the arithmetic unit which receives the incident digits and performs the required operation on them. When this is finished an operation complete pulse is emitted by *M.* and *A.U.* which initiates exactly the same cycle as enumerated in (1).

(3) Transfers from the arithmetic unit to the memory. Here, as in (2), the arithmetic unit prepares itself for operation and then emits a 'record' pulse *via h* which causes the memory to emit 32 clock pulses as before, these cause data to be shifted out of *A.U.* *via k* to *M.*, which now absorbs them in the given position. When the cycle is ended, *M.* emits its operation complete pulse which sends the binary element to its excited state and also causes *A.U.* to emit an operation complete pulse of its own. After this the cycle proceeds as in (1) and (2).

(4) Transfers of the control. In this operation a pulse is emitted by *D.C.* *via* g_3, this passes *via l* to gate the new memory location, as contained in the control register, into the control counter which then contains the new data. After a delay $\triangle$, to enable *C.C.* to settle down, a pulse passes *via c* and initiates the read out operation for the new order exactly as above.

(5) Conditional transfers of control. These depend on the sign of a certain number contained in *A.U.* Thus the decoder emits a pulse to *A.U.* which then emits a standard operation complete pulse if the number is $\geqslant 0$ but emits a pulse on *m* if it is < 0. In the first case the ensuing cycle is exactly the same as follows a standard arithmetic operation of the types described in (1), (2), (3). In the second the cycle is identical with (4).

(6) Input/output orders. If it is imagined that the input/output equipment is in *A.U.* the operation in this case is identical with that described in (1), (2), or (3).

OTHER TYPES OF CONTROL

Those differences which will result from the substitution of a different sort of memory or arithmetic unit are more or less trivial. If a parallel operation memory is used the only change will be that, instead of *M.* emitting 32 clock pulses, it will emit only one which will *simultaneously* gate *all* the digit outputs to the appropriate receptors, this means that *k*, *o* and *p* will become 32 digit channels (or, in general, *n* digit channels).

Similarly, if each order contains the memory location of the next, the only effect will be to eliminate *C.C.*, the pulses on *b* and *d* become one and gate *C.R.* directly to *M.L.* Another set of separate outputs from *C.R.* will be required for the location digits in conditional transfer operations, these will go directly to *M.L.* and will be gated by the pulse on *l*. This is shown in *Figure 5.3.*

6

THE ARITHMETIC UNIT

As its name suggests, the arithmetic unit of a computing machine is that part which operates on stored information to produce the combinations of ordinary arithmetic.

It is customary, in *all* existing computers, to have circuits which will perform the operations of addition and subtraction. Most computers have also a device for multiplication and a minority have division facilities. Such special apparatus as square rooting units are of such limited utility that they have found a place in only one or two existing machines.

In this chapter the logical construction of these and other arithmetic devices will receive consideration, the aim being to provide engineering schematics of the type provided for the control in Chapter 5. These schematics will be in such a form that they can immediately be transformed into actual components by means of the basic circuit units to be described in Chapters 9–12.

Numbers will be assumed to be in true binary notation and to lie in the range $1 > x \geqslant -1$. The form taken by negative quantities is important and two general representations exist. In one, a negative quantity appears as (sign) (absolute value) and in the other as a true complement mod (2). The first system is, perhaps, more suitable for human observation, but has a particular disadvantage from the machine point of view. This lies in the fact that the addition and subtraction of two numbers, represented by (sign) (absolute value), must be preceded by an operation which compares the signs and decides upon the true nature of the operation. Thus:

$$(+)(A) + (-)(B)$$

is really:

$$(+)(A) - (+)(B)$$

and, although the operator has specified an addition, the machine, in fact, performs a subtraction. In machines working in the parallel mode this comparison process may be quite easy, but for serial working, especially if the sign indication appears as usual at the most significant end of the number, it is troublesome. The

following analyses assume complement representation of negative numbers.

MAIN COMPONENTS

Before detailing the design of the various constituent parts of the arithmetic unit it is profitable to examine, in general outline, what must be present.

In the first place an adder or subtractor requires the provision of some form of register in which the sum is to be stored. This may be a register of physically distinct digit storage units of the flip-flop type or alternatively an n pulse delay unit or 'short tank' in which a single number can be stored. Such a unit is absolutely necessary in any form of computer in which two numbers cannot be simultaneously drawn from the store, and it is desirable in all computers.

In the second place a machine capable of multiplication must have an extra register in which the multiplier can be stored. The multiplication of two n digit numbers generates as product a number of $2n$ digits. Since (in most multipliers) the digits of the multiplier are used in sequence there is no reason why the contents of the multiplier register should not be successively displaced to the right and their place taken by the least significant n digits of the product as these are generated. It has been assumed that the product is generated starting from the least significant digit of the multiplier; this is clearly the most advantageous arrangement since the storage register for the last n digits then requires no adding and carrying facilities.

Multiplication requires, then, the two registers mentioned; in any machine in which access to the store is slow, however, it is a great advantage to have a third register in which the multiplicand digits are stored during multiplication. When division is considered the presence of this third register becomes mandatory since the divisor has to be compared with the partial remainder at any stage of the process in order to decide the nature of the next operation to be performed.

It is thus seen that an arithmetic unit should contain three registers capable of holding the digits of interacting numbers. For multiplication it is also necessary that the product and multiplier registers should be such that numbers held in them can be shifted one place to the right on receipt of a suitable impulse. This faculty may also be necessary if the main storage of the machine is of the serial type; the digits of a number then become available one at a time and can be 'staticized' by insertion into such a shifting register.

In machines having automatic division the registers previously mentioned should be able to shift left as well as right to enable partial remainders to be brought to the same size as the divisor.

Before considering the individual arithmetic operations it is profitable to consider the formulation of the shift instructions.

RIGHT AND LEFT SHIFT

It will be shown in Chapter 11 how a register can be constructed to shift its contents in a given direction. Consider first a register capable of right shift, this is equivalent to division of the stored binary number by two. When it is remembered that the numbers concerned may be positive or negative it is evident that such division by two should preserve the sign of the number. If the register stages are $A_1\, A_2 \ldots A_n$ the unit right shift operation should, then, generate the pattern $A_1, A_1, A_2 \ldots A_{n-1}$. To avoid loss of digits the original A_n can then be shifted into the first stage of the multiplier register, if present, and any existing contents of this register similarly shifted, the least significant digit in this register being lost.

The left shift operation is more complex; it would, of course, be possible to construct flip-flop type registers capable of both right and left shift operations but this would be uneconomical; furthermore, in delay type registers, the unit left shift operation, as such, is physically impossible. The solution is to observe that if head and tail of an n-stage register be connected then a single left shift is equivalent to $(n-1)$ right shifts. Here again care must be taken to preserve the sign of the shifted number. Left shift is equivalent to multiplication by 2. For positive numbers left shift will maintain the sign until the number has grown beyond the range of the machine, thus:

$$0{\cdot}01 \ (= +\tfrac{1}{4})$$
$$0{\cdot}10 \ (= +\tfrac{1}{2}) \text{ first shift}$$
$$1{\cdot}00 \ (= -1) \text{ second shift}$$

the second shift not being permissible under the ordinary rules of arithmetic understood by the machine. Again:

$$1{\cdot}01 = (-\tfrac{3}{4})$$
$$0{\cdot}10 = (+\tfrac{1}{2})$$

the shift being unjustifiable since $2 \times -\frac{3}{4} = -1\frac{1}{2}$, which is outside the range of the machine.

On the other hand:

$$1\cdot11 = (-\tfrac{1}{4})$$
$$1\cdot10 = (-\tfrac{1}{2})$$

so that this shift, which is permissible, generates the true signed double of the original number.

ADDITION

Several distinct varieties of adding unit have been described. Perhaps the most common is that compounded from two units known colloquially as 'half adders,' and shown diagrammatically in *Figure 6.1.*

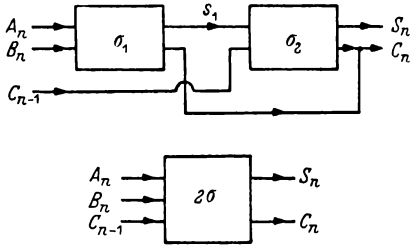


Figure 6.1 (a) Half and full adder

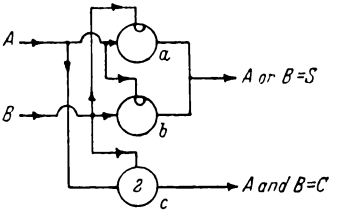


Figure 6.1 (b) Logical construction of half adder

σ_1 and σ_2 are the 'half adder' units; they have the property of combining binary digital inputs applied to A_n and B_n in accordance with the following scheme:

Table 6.1. Half adder matrix for $B_n + A_n$

A_n	B_n	S_1	C_n
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

At any stage in a binary addition process it is necessary to combine up to three digits,

(*a*) A resident or existing digit A_n
(*b*) An incident digit—from store, say, B_n
(*c*) The carry, if any, resulting from a previous addition C_{n-1}

It will be seen that the circuit of *Figure 6.1* performs the operations required to produce a true binary sum and also any carry resulting from the combination of inputs A_n, B_n and C_{n-1}.

Up to this point it has been assumed that the digits to be added were available without any very definite specification of how this was to be arranged. It will now be shown how the device of *Figure 6.1* can be made the basis of several distinct types of addition unit.

First let it be assumed that the digits of A and B are available as simultaneous short pulses separated from other digits of the same numbers by definite time intervals; this situation obtains in delay line type machines. *Figure 6.2* shows how the basic two half adder unit is combined in such a system.

It is assumed that the number B is stored in some form of delay of n digit time duration; A appears, from an external store, one

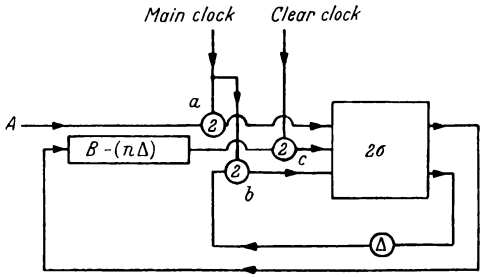


Figure 6.2 Serial adder (delay line type)

digit at a time starting from the least significant and in synchronism with the digits of B. A single pulse time delay Δ arranges for the carry C_p, from the pth addition stage, to be available to the adder at the time required for input to digit addition $(p+1)$. The gates marked a and b are arranged to cut off the incident and carry digits except at times dictated by the arrival of suitable 'clock' pulses; this enables any particular number to be selected from the external store and added to that already contained in $(n\triangle)$.

A matter frequently overlooked in the examination of a device of the type described above is the fact that it is impossible to clear data out of an n digit delay (of the acoustic type) without waiting for a period of n digits. Allowing the necessity for this, there still remain several alternative methods, for example:

(i) A separate 'clear' operation can be included.

(ii) A 'read out *and* clear' order can be included in addition to the normal read out.

(iii) The clearing can be accomplished at the same time as the next addition.

Alternative (i) would waste one complete word time and is therefore undesirable. Alternative (ii) requires that the problem analyst remembers to insert the appropriate order *before* the actual operation requiring the clear line is considered. This obviously offers scope for error. Alternative (iii) therefore appears to be the most satisfactory. From the engineering standpoint there is little difference, since all of the above variants can be constructed with the aid of a single gate c. For (i) gate c receives clock pulses at all times *except* when clearing, for (ii) at all times *except* when reading out *and* clearing, and for (iii) at all times *except* when adding into cleared line.

A second type of serial adder employs a storage shifting register for the number B. This is characterized by the fact that the digits of

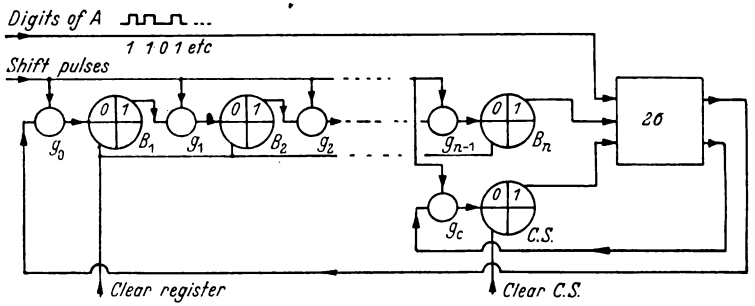


Figure 6.3 Serial adder (shifting register type)

B are static but can be shifted, one place at a time, to the right on receipt of clock pulses. An addition unit constructed on this plan is shown in *Figure 6.3*. The number B is stored in the register stages $B_1 \ldots B_n$; digits from the store and from B_n are continuously incident on 2σ but nothing occurs, except when the shift gates $g_1 \ldots g_n$ of the register and the auxiliary gates g_0 and g_c are pulsed. When this happens, the sum of incident, resident and carry (stored in *c.s.*) digits is shifted, *via* g_0 into B_1; B_n is lost and replaced by B_{n-1} *etc.* and the carry for the next stage (if any) is shifted into *c.s.* Since a shifting register permits access to all digits of B simultaneously, clearing can be accomplished as a one shot operation prior to addition if required. Similarly, the carry store *c.s.* is *always* cleared prior to addition unless multiple length addition is required.

Finally, it is shown in *Figure 6.4* how the 2σ units can be arranged to provide parallel addition of a number of channels.

$A_1 \ldots A_n$ and $B_1 \ldots B_n$ are the respective stages of the A and B storage registers. Addition takes place in the units $2\sigma_1 \ldots 2\sigma_n$

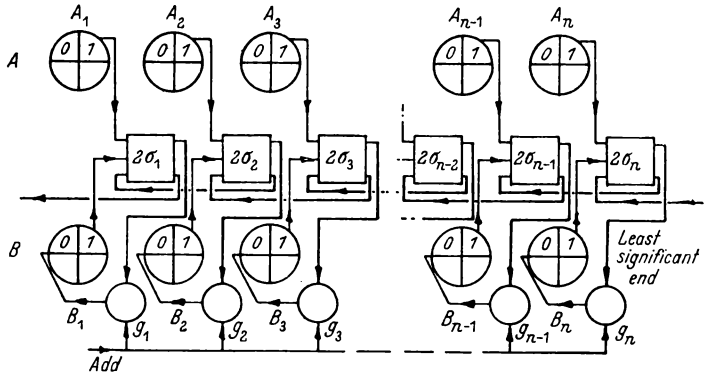


Figure 6.4 Parallel adder (register type)

interconnected so as to propagate the carry. When the circuit has settled down a pulse is applied to all of the gates $g_1 \ldots g_n$ simultaneously causing the sum of A and B to be sent to B. It should be noted that the shift pulse is assumed to be sufficiently short,

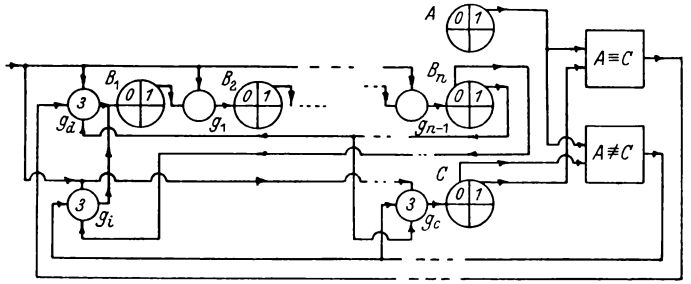


Figure 6.5 Serial adder

compared with the response time of the 2σ units to changes at their inputs, for B to change without affecting the inputs to the gates $g_1 \ldots g_n$.

In the case of a register type machine, a completely different adding scheme is available, this depends upon the fact that the

contents of B_n (*Figure 6.5*) need not be known to the addition circuit as such. The scheme of operation is as follows:

If $A_m \equiv C_m$ send B_m to B_1 and shift one place to right; do not alter C_m

If $A_m \not\equiv C_m$ send inverse of B_m to B_1 and shift one place to right, sending B_m to C_m

Here the symbols A_m, B_m and C_m represent the contents of A_n, B_n and C_n after m operations. The circuitry involves only standard coincidence and anti-coincidence units and three input gates g_i, g_d and g_c.

Operation depends upon the fact that if

$$A_m \equiv C_m \quad \text{then sum digit } (A_m + C_m) = 0 \quad \text{carry } (A_m + C_m) = C_m$$

and if $A_m \not\equiv C_m$ then sum digit $(A_m + C_m) = 1$ carry $= B_m$

which is readily seen from the following table.

Table 6.2

A	*C*	*B*	*Sum*	*Carry*
0	0	0	$0(=B)$	$0(=C)$
0	0	1	$1(\ ,,\ )$	$0(\ ,,\ )$
1	1	0	$0(\ ,,\ )$	$1(\ ,,\ )$
1	1	1	$1(\ ,,\ )$	$1(\ ,,\ )$
1	0	0	$1=(\neq B)$	$0(=B)$
1	0	1	$0\ (\ ,,\ )$	$1(\ ,,\ )$
0	1	0	$1\ (\ ,,\ )$	$0(\ ,,\ )$
0	1	1	$0\ (\ ,,\ )$	$1(\ ,,\ )$

Input from the store is at A; upon receipt of shift pulses, coincident with the digits of the word which it is required to add, register $B_1 \ldots B_n$ performs right shift operations and B_n is fed either direct *via* g_d or inverted *via* g_i into B_1. At the same time B_n is fed directly to C *via* g_c if g_i is actuated. The anti-coincidence unit is identical with its coincidence mate except that its connection to C is inverted.

An entirely different class of adding element is the diode network. This is in reality a combination of a many-one and a one-many function table and will not be described further here, its mode of operation will be evident from the discussion of Function Tables contained in Chapter 11.

Proposals have appeared, from time to time, for the construction of special valves for the operation of binary addition. The best known of these is due to J. Katz, who has devised a series of such

devices which would be particularly suitable for incorporation in a parallel accumulator. By deflecting an electron beam in the electrostatic field of three electrodes, the combinations required to produce sum and carry digits are generated in a single valve which, it is stated, requires only a few milliamperes of current at a potential of 50 V for operation. Unfortunately these components are not, at present, available.

SUBTRACTION

Two distinct methods are available for the design of a subtraction unit; in the first a 'half subtractor' unit is constructed to have the properties shown in *Table 6.3.*

Table 6.3. Half subtractor matrix for $(B_n - A_n)$

A	B	S_n	C_n
0	0	0	0
1	0	1	1
0	1	1	0
1	1	0	0

This results in a subtractor unit which is identical in layout with the adder unit of *Figure 6.1.* The logical construction for a half subtractor unit is shown in *Figure 6.6.*

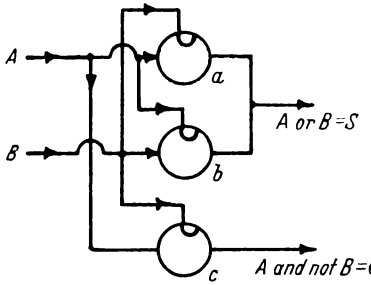


Figure 6.6 Logical construction of half subtractor

In this figure a second gate has been used to obtain the carry, in conformity with *Figure 6.1 (b)*; it would, however, have been equally possible to buffer the output from gate *a* and to use this.

The second type of subtractor makes use of the fact that subtraction of *A* from *B* is equivalent to the addition of the *true* complement

of A to B. This can be readily achieved by presetting the carry store to unity (the 'elusive 1' of computing parlance) and adding the *digital* complement of A to B.

Since the digital complement of a binary number is obtained by changing 0's into 1's and *vice versa* this is trivially derivable from the incident digit stream.

MULTIPLICATION

The process of multiplication has frequently been engineered into computing machines in such a way as to permit of operation only when both multiplier and multiplicand (and consequently also the product) are positive; this has been due, to a large extent, to the detailed following of the Goldstine–von Neumann scheme of correction of product when interacting numbers are non-positive. In essence, all of the available schemes of multiplication for non-positive numbers are complicated by the fact that negative numbers are invariably stored in complement form. If numbers were stored as (sign) . (absolute value) multiplication and division would be as easy for non-positive numbers as for the simpler case. On the other hand, as was pointed out in an earlier section, addition and subtraction—which are the most used operations—would be more complex.

Supposing, then, that negative numbers are stored as complements (Mod. 2), the following cases then arise:

$$\text{(i)} \quad +a \times +b = +ab$$

$$\text{(ii)} \quad +a \times -b = [+a \times (2-b)] = 2a - ab$$

$$\text{(iii)} \quad -a \times +b = [(2-a) \times b] = 2b - ab$$

$$\text{(iv)} \quad -a \times -b = [(2-a)(2-b)] = 4 - 2a - 2b + ab$$

Thus, after a straightforward multiplication case (i) would be correct, (ii) would require the subtraction of $2a$, (iii) the subtraction of $2b$ and (iv) the subtraction of $(4-2a-2b)$. Whilst it is true that Goldstine and von Neumann have shown how at least some of these corrections can be applied during the course of the multiplication, their scheme is not well adapted to the case of a serially operating machine in which the signs of numbers are not continuously available.

An alternative scheme, suggested by the present authors, does not involve any knowledge whatever of the signs of the interacting numbers. The scheme of operation, for binary numbers, is as follows: Let a_n and b_n be the nth digits of A and B respectively.

Then the following process will generate the correct product $A \times B$ whatever the signs of A and B.

$n \neq 1$ (1) If $b_n = b_{n+1}$ Shift partial product already obtained one place to right.

(2) If $b_n = 1$, $b_{n+1} = 0$ Subtract A from partial product and shift latter one place to right.

(3) If $b_n = 0$, $b_{n+1} = 1$ Add A into partial product and shift latter one place to right.

Repeat until $n = 1$

$n = 1$ Exactly as above except that the shift is deleted. Observe that initially $b_{n+1} = 0$.

The following is a constructive proof of the validity of the process. Let a_{n-p+1} be the contents of the accumulator after the pth stage of the process and let m represent the multiplicand. Then the following recursion formula expresses the above process:

$$a_{n-p} = \tfrac{1}{2}a_{n-p+1} + (b_{n-p+1} - b_{n-p}).\, m \qquad [p = 1 \ldots (n-1)] \quad \ldots . \quad (6.1)$$

It will be observed that the factor $(b_{k+1} - b_k)$ has the values:

$$\begin{aligned} 0 &\quad \text{if } b_k = b_{k+1} \\ 1 &\quad \text{if } b_k = 0,\ b_{k+1} = 1 \\ -1 &\quad \text{if } b_k = 1,\ b_{k+1} = 0 \end{aligned}$$

as required by the description.

Since, at the first stage, the convention $b_{n+1} = 0$ holds and the accumulator is clear:

$$a_n = 0 + (b_{n+1} - b_n).m = -b_n m \qquad \ldots . \quad (6.2)$$

Thus, multiplying the pth member of equation 6.1 by 2^{p-1}:

$$\begin{aligned} a_{n-1} &= \tfrac{1}{2}a_n + (b_n - b_{n-1}).m \\ 2a_{n-2} &= a_{n-1} + 2(b_{n-1} - b_{n-2}).m \\ 2^2 a_{n-3} &= 2a_{n-2} + 2^2(b_{n-2} - b_{n-3}).m \\ &\ldots\ldots\ldots\ldots\ldots \\ 2^{n-2}a_1 &= 2^{n-3}a_2 + 2^{n-2}(b_2 - b_1).m \end{aligned}$$

whence, adding:

$$2^{n-2}a_1 = \tfrac{1}{2}a_n + [-2^{n-2}b_1 + 2^{n-3}b_2 + \ldots + b_{n-1} + b_n].m$$

or, eliminating a_n by means of equation 6.2, and dividing through by 2^{n-2}

$$a_1 = \left[-b_1 + \tfrac{1}{2}b_2 + \ldots + \frac{1}{2^{n-1}}b_n \right].m \quad \ldots . \quad (6.3)$$

Now if b is positive $b_1 = 0$ and $\sum_{p=2}^{n} \frac{1}{2^{p-1}} b_p = b$ and similarly, if b is negative $b_1 = 1$ and $\sum_{p=2}^{n} \frac{1}{2^{p-1}} b_p = (1 - |b|)$ whence, for b negative, $\left[-b_1 + \sum_{p=2}^{n} \frac{1}{2^{p-1}} b_p \right] = [-1 + (1 - |b|)] = -b$ and it is seen that equation 6.3 can be written:

$$a_1 = b.m \quad \ldots . \quad (6.4)$$

whatever the sign of b. Thus, since no restriction was placed upon the sign of m, the process gives a correctly signed product for all sign combinations.

The logical design of a multiplier constructed on the above basis is shown in *Figure 6.7.*

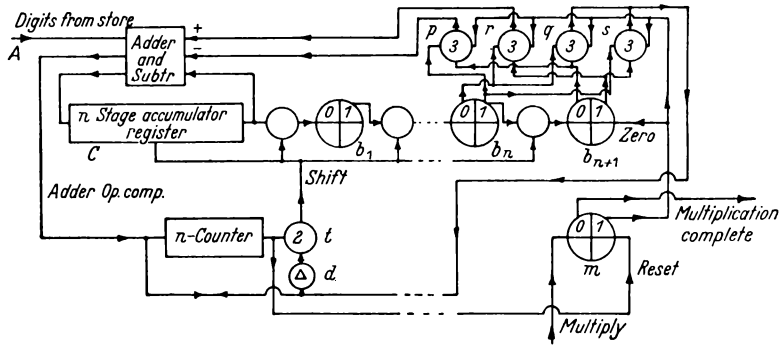


Figure 6.7 Multiplier

The nth and $(n+1)$th stages of the B register are b_n and b_{n+1}. The operation of multiplication is signalled by the triggering of the setting flip-flop m from the function table. When this occurs, b_{n+1} is set to zero and gates p, q, r and s are tested; if $b_n = b_{n+1}$ either q or s will react and produce an output which operates the n counter. Assuming that the last operation has not been reached,

b_1, the whole contents of B are shifted, b_n is replaced by b_{n-1} and b_{n+1} by b_n.

Should any change have taken place in the relative contents of b_n and b_{n+1} one or other of the gates p, q, r, s will again react. For $b_n=0$, $b_{n+1}=1$ r emits a pulse to the adder and for $b_{n=1}$, $b_{n+1}=0$ p emits a pulse to the subtractor. These units cause A to be added or subtracted from C and then emit a signal that their operation is complete. This operation complete signal actuates the counter exactly as before and, if necessary, initiates a shift of C and B as before.

At the last operation the n counter will close gate t, thus inhibiting the shifts, and signal the end of the multiplication by resetting m.

DIVISION

The process of division in the case of a binary based computer has marked similarities to that of multiplication. Specifically, there are two alternative processes for its operation:

(1) The standard trial-and-error method.
(2) The so-called non-restoring method.

In the first of these the divisor is compared with the partial remainder at any stage (the numbers being assumed positive). If it is *less* than the partial remainder a subtraction is performed and unity placed at the right hand end of the partial quotient; if, on the other hand, it is greater than the remainder, no operation is performed and zero is placed at the right hand end of the partial quotient. The partial remainder is then doubled and the process repeated to give a quotient of any desired precision. This method suffers from two disadvantages, firstly that it will work only for positive numbers, and secondly that the only simple method of comparing the relative size of divisor and partial remainder is to perform a subtraction. This subtraction is redundant when the remainder is less than the divisor and requires (in most machines at least) a subsequent addition to annul it.

The non-restoring method is considerably more flexible since it will operate on numbers of any sign and does not require the performance of any unnecessary operations.

Assuming that the divisor (m) is stored in the memory, the

dividend (d) in the accumulator register and that the quotient (b) appears in a shifting register the process is as follows:

Operations $1-(n-1)$	
If $m(1)=a(1)$	Add $2^{-(n-1)}$ to (b)
	Shift left accumulator and register one place
	Subtract m from a
If $m(1)\neq a(1)$	Add 0 to (b)
	Shift left accumulator and register one place
	Add m to a
Operation (n)	Add $2^0+2^{-(n-1)}$ to (b)

In this scheme $a(p)$, $m(p)$ and $b(p)$ represent the pth digits of a, m and b.

A constructive proof is as follows: Let r_k = remainder after the kth operation and let b_k be the kth digit of b. Then the recursion formula for the above process is:

$$r_k = 2r_{k-1} + [1-2b_k].m \qquad \ldots\ (6.5)$$

Notice that

$$r_0 = a \qquad \ldots\ (6.6)$$

Multiplying the terms of equation 6.5 by appropriate powers of 2 it is seen that:

$$\begin{aligned}
r_{n-1} &= 2r_{n-2} + [1-2b_{n-1}].m \\
2^1 r_{n-2} &= 2^2 r_{n-3} + 2^1[1-2b_{n-2}].m \\
2^2 r_{n-3} &= 2^3 r_{n-4} + 2^2[1-2b_{n-3}].m \\
&\cdots\cdots\cdots\cdots \\
2^{n-2} r_1 &= 2^{n-1} r_0 + 2^{n-2}[1-2b_1].m
\end{aligned}$$

whence, adding:

$$r_{n-1} = 2^{n-1} r_0 + m\sum_{p=0}^{n-2} 2^p - m\sum_{p=1}^{n-1} 2^{n-p} b_p \qquad \ldots\ (6.7)$$

or, using equation 6.6, dividing by $2^{n-1}.m$ and rearranging:

$$a/m = \frac{r_{n-1}+m}{2^{n-1}.m} - 1 + [b_1 + \tfrac{1}{2}b_2 \cdots \frac{1}{2^{n-2}} b_{n-1}] \qquad \ldots\ (6.8)$$

(since $\sum_{p=0}^{n-2} 2^p = 2^{n-1}-1$).

It follows that the scheme given above will generate the true quotient a/m whatever the signs of a and m and except for the sign digit, to correct which unity must be added to the result. The

reason for the addition of the final unit ($2^{-(n-1)}$) lies in the fact that this gives the least biased round-off when b_{n+1} is unknown.

It is particularly easy to construct the logical diagram for division using the non-restoring method.

$d_1 \ldots d_n$ is the divisor register, $a_1 \ldots a_n$ the partial remainder register and $b_1 \ldots b_n$ the quotient register. For simplicity it is assumed that the a and b registers are capable of direct left shift.

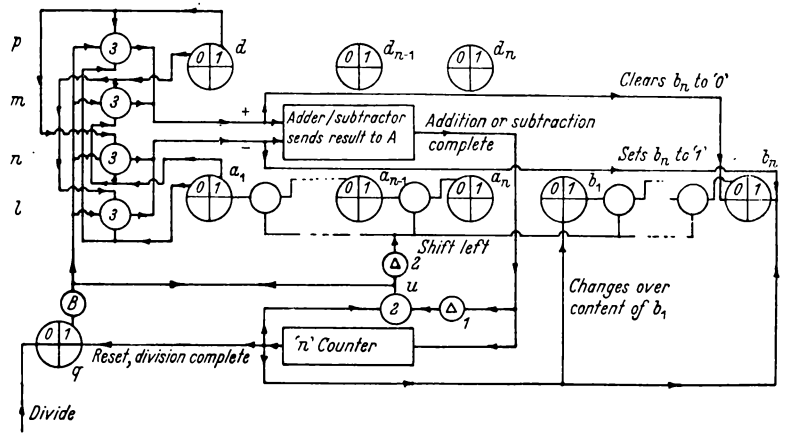


Figure 6.8 Schematic of binary divider

Sign digits d_1 and a_1 are compared, *via* the gates l, m, n, p. When the divider is switched on a flip-flop q is set up; this emits, *via* buffer B, a pulse to test gates l, m, n, p. Gates l and n emit a pulse when d_1 and a_1 have the same sign and thus call for subtraction of d from a, at the same time unity is sent to quotient register stage b_n. Similarly, if d and a have different signs one or other of gates m and p emit the delayed B test pulse to call for addition, at the same time b_n is set to zero. Meanwhile, the pulse from B has passed, *via* a delay $\triangle_2$, and has caused a left shift of the contents of a and b. It is assumed that there is sufficient delay in the commencement of the add or subtract operation for this shift to be complete before the former begins.

When the addition or subtraction is over an operation complete pulse is emitted. This operates the n counter and, after a delay $\triangle_1$ to enable the latter to settle down, tests gate u. If the division is not complete u passes the test pulse to $\triangle_2$ and causes a and b to shift one place left, this shifts b_n into b_{n-1}, and effectively multiplies the partial remainder, in a, by 2. Gates l, m, n, p are again tested and the cycle repeated.

When the required number (n) of add/subtract operations is complete the n counter will close gate u, reset division flip-flop q and cause stage b_1 to 'count,' i.e. invert its contents, whilst b_n is set to unity. Flip-flop q then emits the overall operation complete signal to the remainder of the machine.

DECIMAL AND OTHER ARITHMETIC

So far it has been assumed that the computing machine under discussion operates with numbers represented in true binary form. This is the case in almost all of the machines constructed to date, and is dictated by the greater simplicity of machines operating in this scale. There is no difficulty, in principle, in designing a computer to work in any scale whatever and the adding and multiplying units of such a device will now be considered.

One means of representing a number, to base n, and having p digits, is to let each digit occupy $(n-1)$ units of time. Thus, in decimal scale, the number 981 might appear as:

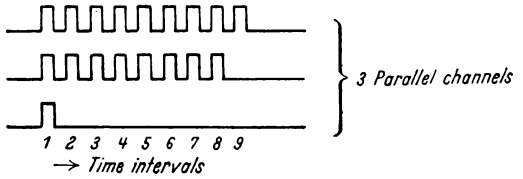


Figure 6.9

where some form of pulse on each channel represents the digit. This corresponds to a parallel operation system. Alternatively, in serial working, the same number could be written:

Figure 6.10

where the same number now occupies three times as great a time interval as before. Or, yet again, 30 channels might be used, only three of which (corresponding to 9, 8 and 1) receive impulses at a given time.

A completely different method of representation is to code each digit in terms of, say, binary scale.

Again three representations are possible and, taking the number 981, as before:

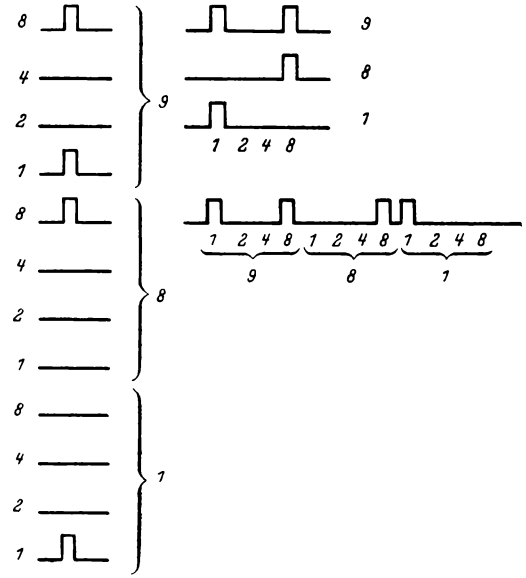


Figure 6.11

In the first all digits are available at the same instant of time—at the expense of 12 channels. In the second four time intervals are needed but the number of channels is reduced to three, and in the third, or truly serial, mode of working, twelve time intervals are required on a single channel.

At this point it becomes clear that the equipment, or time requirement, is less in binary coded representation than in straightforward decimal notation. Even further saving results when the true binary form of the number is used, thus for the previous example:

$$981 = (1001)(1000)(0001)$$

in the *coded* notation, whilst

$$981 = 1111, 0101, 01$$

in *true* binary notation, a saving of two channels or two time intervals.

The logic of adding circuits in non-binary scales is directly related to the representations discussed above. The binary element, or flip-flop, is no longer the unit element and some form of

n counter must be used for each digit. Two general forms of this unit exist, in the first a set of on-off elements are connected into a ring, as shown in *Figure 6.12.*

Here the elements are represented by $b_0 \ldots b_{n-1}$ and are so arranged that only one at a time can be in the *on* state. On receipt of impulses the *on* condition is advanced one stage at a time; it follows that if the lth element of such a device is *on* and m impulses are applied the final state is $(l+m)$ [mod n]. To construct a multi-digit adder from these units, it is merely necessary to arrange that

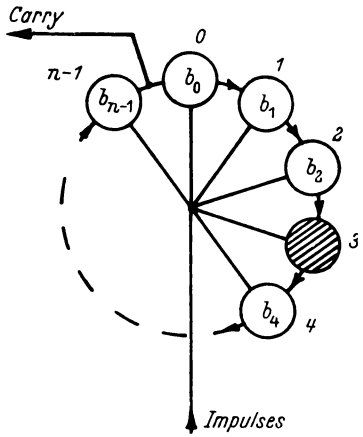


Figure 6.12 Ring counter

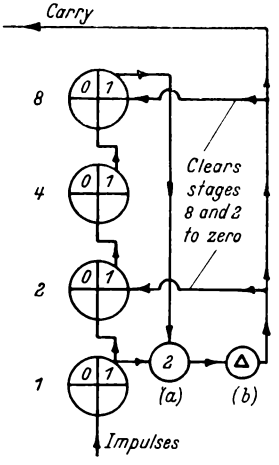


Figure 6.13 Binary counter with reset at ten

when the *on* condition passes from element b_{n-1} to b_0 an impulse is sent, *via* a carry lead, to the next ring of the set.

Naturally it is necessary to ensure that the propagating carries do not interfere with any set of digits which are being sent to other rings, and to ensure this it is usual to store all carries until the end of the individual digit additions and then to propagate them, one at a time, starting from the least significant.

The second form of n counter in general use is some form of binary scaler. For the decimal scale, as was seen above, four binary stages per decimal digit suffice and incident pulses cause these to make a binary count in the normal manner. Since the modulus of operations is (in the 4 binary stage counter) 16 it follows that when the sum of the counter contents and the incident digit exceeds 9 an incorrect representation will occur and no carry is

necessarily produced. To correct this two techniques have been used; in the first, shown in *Figure 6.13*, an auxiliary gate *a* is primed when the counter reaches 8 and then emits the advancing pulse from the first stage (corresponding to the transition 9→10). This pulse is delayed by *b* to allow the second counter stage to settle down and is then applied to stages two and four forcibly to reset them to zero, at the same time this reset pulse is passed on to the next counter as a carry. In the second form of correction the digits of the two numbers are added as usual and then 6 is added to each of the counters. It will be seen that this results in the emission of a carry impulse from those stages whose contents lie between 10

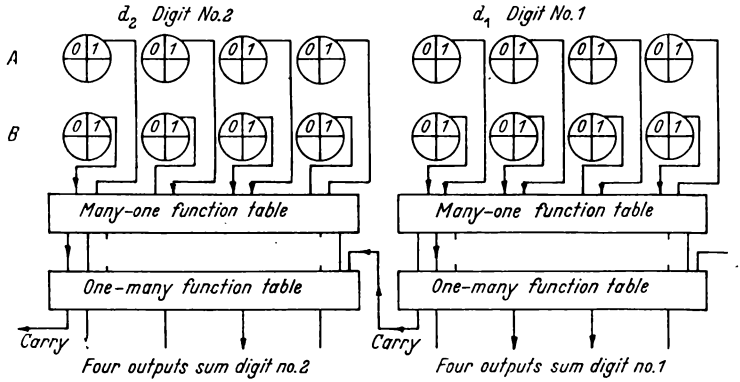


Figure 6.14 Parallel decimal function table adder

and 15 and the correction of their contents to its value mod (10). (For example $14 = 10 + 4$ and $14 + 6 = 16 + 4$.) This method is not without its disadvantages since means must be provided to restore any counter, from which a carry has *not* arisen after the addition of 6, to its original state.

An entirely different form of adder is applicable when the binary coded digits of the two numbers to be summed are available simultaneously in, for example, a pair of registers. In the case of decimal numbers the sum may be generated by the arrangement shown in *Figure 6.14*.

In this type of adder the groups of four binary digits, corresponding to the decimal digits of the numbers to be summed, are applied to sets of function tables. First a many-one function table generates outputs on twenty lines corresponding to the possible digit sums 0 to 19; this is followed by a one-many function table which

reduces this single output to a correct sum digit, mod (10), in coded binary form on four outputs together with a carry if required. This carry is passed on to the next many-one function table of the series. The arrangement has the advantage of providing a static output, but is very costly in equipment.

The complexity of the circuits required for addition in non-binary scales will have prepared the reader for the fact that multiplication is even more cumbersome. The simplest type (conceptually at least) of multiplier follows closely the technique which is customary with ordinary desk multiplying machines of the Brunsviga variety. The principle is simple; if A is to be multiplied by B an accumulator is arranged so that, if the digits of B are $b_n \ldots b_1$, A is added into this unit b_1 times, shifted to the *left* one place, added in b_2 times and the process continued until all of the digits of B have been treated. The accumulator then contains the product $A \times B$. A variant of this process keeps the digital position of A fixed and shifts the partial product digits, in the accumulator, to the *right*.

The above process will only work when all digits of A and B are represented in the same scale. For example, £ s. d. cannot be operated upon.

More sophisticated techniques for the multiplication of non-binary numbers have been devised but, for electronic machines at least, have not been much used. Two examples will suffice to illustrate some of the principles which are involved.

First it may be observed that the successive addition and shift multiplier described above may take up to $(p-1)n$ addition times to generate the product of two n digit numbers to base p. An improvement on this situation can be effected by the use of built-in multiplication tables. A schematic for the process is given in *Figure 6.15.*

The digits of the number A are held in stages $A_n \ldots A_1$ (the nature of the A_n units will not be specified more closely as the reader will be aware of their construction from the preceding discussion of adders). The digits of B are held in stages $B_n \ldots B_1$ of a register so arranged that the contents can be shifted, one place at a time, in the direction $B_n \rightarrow B_1$. Each digit of A is multiplied by B_1 by means of a set of function table units, and the outputs of these, which may consist (for decimal scale at least) of up to two digits, are summed together in the matrix adders S_1 which are so interconnected as to form the correct output $B_1 \times A$. The digits of this product are now added to those of the contents of the product register $P_n \ldots P_1$ by means of the set of matrix adders S_2. The sum so formed is then transferred to the product register

$P_n \ldots P_1$ but shifted one place to the *right*. To avoid loss of data as the multiplication proceeds, the extreme right-hand digit of the partial sum is always shifted into B_n and the contents of B similarly shifted. In this way, by repeating the process just described, the contents of B_1 are successively $B_1, B_2 \ldots B_n$ and at the end of n cycles of the process the product $A \times B$ is found to be stored with its first n digits in $P_n \ldots P_1$ and its last n digits in $B_n \ldots B_1$.

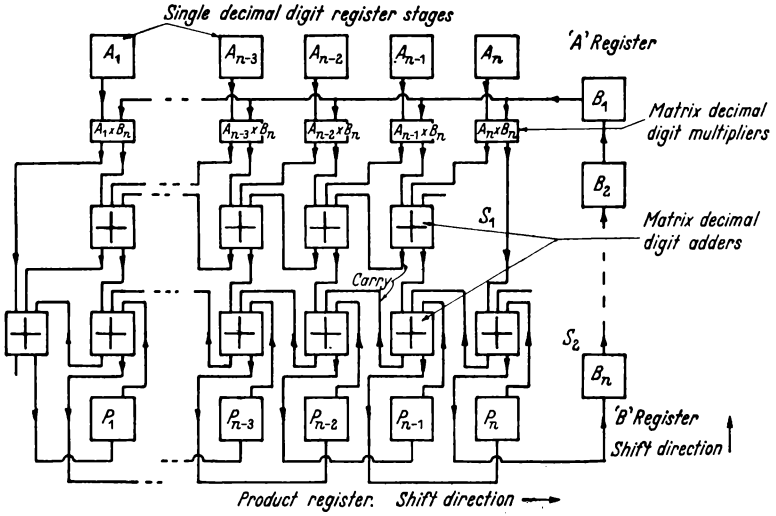


Figure 6.15 Schematic of single cycle decimal multiplier

By suitable attention to registers and matrices the above process can be made to deal with numbers of non-uniform scale. The multiplication time with this type of device is only $2n$ addition times but, as will be readily seen, the cost in components is likely to be high.

To conclude the discussion of multipliers the so-called 'halving and doubling' process will be described. This depends upon the fact that:

$$a \times b = (2a).\left(\frac{b}{2}\right) \text{ if } b \text{ is even}$$

$$= (2a).\frac{(b-1)}{2} + a \text{ if } b \text{ is odd}$$

it follows that, in order to obtain the product of two numbers, it is sufficient to repeat the above process until b is reduced to unity,

and then to add the various remainders together. The following example illustrates the process:

$$
\begin{aligned}
& 981 \times 314 \\
= \; & 1{,}962 \times 157 \\
= \; & 3{,}924 \times 78 + 1{,}962 \\
= \; & 7{,}848 \times 39 + 1{,}962 \\
= \; & 15{,}696 \times 19 + 1{,}962 + 7{,}848 \; (= 9{,}810) \\
= \; & 31{,}392 \times 9 + 9{,}810 + 15{,}696 \; (= 25{,}506) \\
= \; & 62{,}784 \times 4 + 25{,}506 + 31{,}392 \; (= 56{,}898) \\
= \; & 125{,}568 \times 2 + 56{,}898 \\
= \; & 251{,}136 \times 1 + 56{,}898 \\
= \; & 308{,}034.
\end{aligned}
$$

The mathematically inclined reader will have noticed that the process is effectively the conversion of b into its binary equivalent and the multiplication of a thereby. It is also true that this algorithm will generate the correct product whatever the scales of notation of the two numbers, and this makes clear the reason for the adoption of the method in certain business machines.

To construct a multiplier using this technique is a simple exercise in the methods discussed in this chapter and will not be considered in greater detail, save to remark that if the digits of the numbers are given in binary coded form the multiplications and divisions by two are particularly easy.

To conclude this chapter it may be mentioned that the non-binary dividers suggested up to the present are more or less identical in principle with the ordinary process taught in the schools. The particularly elegant non-restoring process described for binary numbers is not applicable in any other scale of notation.

7

MISCELLANEOUS OPERATIONS

In addition to the normal arithmetic functions mentioned in the last chapter, it is necessary for an automatic calculating machine to have in its repertoire certain other operations of a non-arithmetic character.

Clearly such instructions as: 'Record number in register in memory location (x)' are necessary to the working of the machine; their existence was clearly envisaged in Chapter 4, and their logical construction is so much an integral part of the construction of the store that it will be held over until Chapter 12 where the physical details of this organ receive consideration.

Less obvious is the need for an operation of the type: 'Record the first n digits of the number in the register in the first n digital positions of memory location (x), leaving the remaining contents of the latter unchanged.' The special use of such an instruction is as follows; suppose that the standard instruction word is of the following form:

> 'Perform operation A on number stored in memory location (x) and go to location (y) for the next instruction'

and that the digits specifying A, (x) and (y) are arranged in the standard word as follows:

Location	Application
$1-n$	(x)
$(n+1)-m$	A
$(m+1)\rightarrow(m+1+n)$	(y)

The effect of the modified record instruction is to enable the value of (x) in any other instruction to be changed to some other value $(x)^1$ say. This process has an important application in allowing the machine to make use of tabulated functions stored in its memory, thus suppose that the function $\sin\frac{\pi}{2}(x)$ is stored in the memory in such a way that x varies in steps of 2^{-n} and that the sine values are

stored in memory locations having numbers x. Now assume that during a calculation it is required to find $\sin \frac{\pi}{2}z$ where z is generated as a result of the activities of the machine and is initially unknown to the person preparing the problem for machine calculation. All that need be done is to insert, at the appropriate place, y_1 say, an instruction 'Contents of memory location (?) to register, *etc.*' The place marked (?) is left blank and when z has been generated in the register by the machine the partial substitution order: 'Record first n digits of register in first n digital positions of memory position y_1 *etc.*' is used. This now inserts into the original order the value z, so that the order now reads effectively:

$$\text{'Sin } \frac{\pi}{2}z \text{ to register, } \textit{etc.}\text{'}$$

The second non-arithmetic order which needs consideration is the so-called 'collate.' One form of this is as follows: 'Compare the number in register R with that in memory location (x) and place in register A a number having 1's wherever R and (x) have 1's and zeros elsewhere.'

This instruction has uses very similar to those of the partial substitution order discussed above. For example, the problem of finding $\sin \frac{\pi}{2}z$ might have been solved by sending the instruction to register A after collating with a number consisting of n zeros followed by a string of 1's—this would remove any previous z value—then *adding* the value of z into register A, and re-recording register A contents in the original memory location. This process is somewhat more complicated than that involving the partial substitution order; however, the collate instruction has other uses to which the latter does not apply. For example, suppose that it is desired to split the accumulator register into m independent sub-registers in order to deal efficiently with numbers of much less precision than those normally used by the machine. Clearly, if each small number was separated from its neighbours by a vacant position or zero, a single addition of two such numerical groups could not produce carries which interfered with other such groups. This does not apply, however, to subsequent additions and it is necessary to clear the zero spaces prior to each such operation. This clearing can be easily done by collation with a number consisting of 1's everywhere except at the positions of the original spacing zeros, at which positions zeros are placed; the result of this

collation will be to leave the sub-sums unaltered, but to clear out the space positions.

Other applications of collation include the comparison of given digits of two numbers and the use of the machine for mechanical translation.

THE PARTIAL SUBSTITUTION

The logical design for this instruction depends largely upon whether the mode of operation of the machine is serial or parallel. For a serial machine the circuitry of *Figure 7.1* is adequate.

The number to be recorded is held in register R, flip-flop p is set up; this calls for emission of suitably placed clock pulses from the store and opens gate q to admit them to the n counter and to the register shift gates $g_1 \ldots g_n$. The counter is normally preset

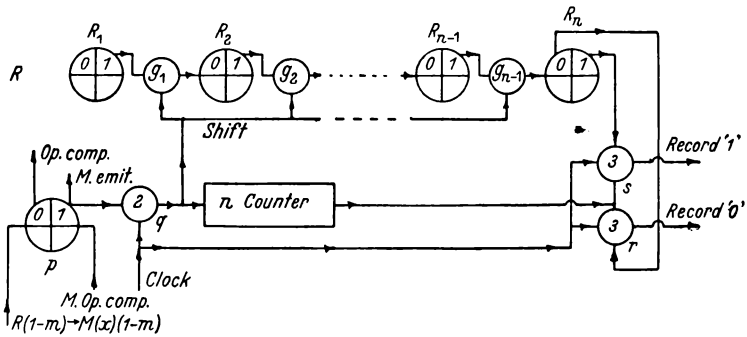


Figure 7.1 Partial substitution—serial

to the complement of n so that when n digits remain in R, gates r and s are opened to the clock pulses which then cause recording of the appropriate digits of R to occur. At the end of the operation p is reset by the memory operation complete pulse and in turn emits an operation complete pulse to the rest of the machine.

In parallel operation type machines the partial substitution instruction is trivial since it involves only the suppression of the record gates on the unwanted channels.

COLLATION

A suitable collation circuit for serial machines is given in *Figure 7.2*. The number being collated is emitted from the store under the command of flip-flop c. The collation number is stored in

register R which has its head and tail elements connected by gate g_0. When marker clock pulses arrive from the memory they pass gate d and cause registers R and A to execute right shifts. Gate e transmits 1's to register A when both the input digit from the store and the last stage contents of register R are unity and zeros other-

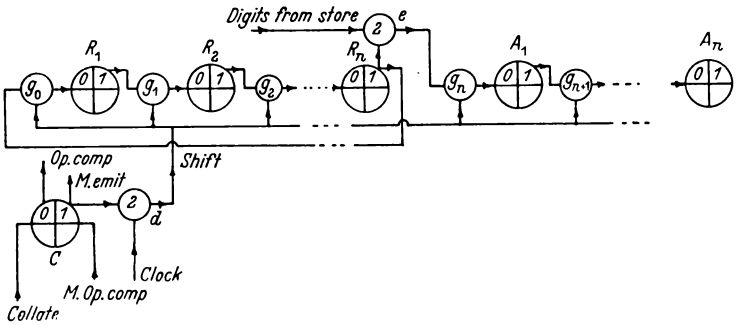


Figure 7.2 Collation

wise. Thus at the conclusion of the operation the collation number remains in register R (having been recycled by gate g_0) and the collation result is found in register A. The memory operation complete pulse resets c thus closing d and emitting an operation complete pulse to the rest of the machine.

8

INPUT—OUTPUT

THE subject of input–output instructions for an automatic computing machine is so bound up with the nature of commercial equipment that it will be necessary, in this chapter, to relate the discussion to available apparatus. This is appropriate since previous chapters have been logical in context, whereas subsequent chapters will deal with engineering aspects of the various organs, and the subject-matter of this section can be thought of as a bridge between the two sections.

It would be impossible, in the space available, to detail every variant of each type of input or output equipment; instead a typical example of each general variety will be examined.

Input devices can be classified into:

(1) Key switches.
(2) Teleprinter keyboards.
(3) Punched tape sensed electrically, electronically or photo-electrically.
(4) Punched cards read as in (3).
(5) Film read photo-electrically.
(6) Magnetic tape and slow drums.
(7) Direct character sensing from written or printed page.
(8) Possible voice-operated equipment.

For output, there are available:

(1) Filament or neon lamps.
(2) Teleprinters and teleprinter tape punches.
(3) Multi-channel punched card producers and reproducers.
(4) Magnetic tape or slow drums.
(5) Film produced from (1).
(6) Film produced from cathode ray tube display.
(7) Possible audio output.

These two classes of equipment will be taken in order and their application to a computing machine considered. The general principles to be observed are, that any input device should be capable of keeping the machine supplied with data at a speed

sufficient to prevent wasted time, and that any output device should be fast enough to avoid disproportionate loss of machine time during the output operation.

INPUT DEVICES

Whereas most extant calculating machines have a manual input on key switches, these cannot be considered adequate. For solving a simple problem, of the type 'find $\sqrt{x}$,' where the routine for solution is *already* stored in the memory they are, of course, as good as any other device since x would have, in the last analysis, to be first presented to *any* input device by either writing or key depression. When the computing routine is not already in the machine, however, such an input device is quite inadequate, since the time of insertion of programme information would be, for all existing and for probably all future machines, of the order of 15 minutes. This does not imply that such an input should be omitted since it is particularly suitable for test purposes.

The teleprinter keyboard suffers similar disadvantages to the key switch. It is perhaps slightly faster and has the advantage of preparing a printed record, but unless the calculating machine has a store of such a size that all programmes ever used by the machine are retained permanently it cannot be considered satisfactory.

Teletype equipment can produce a permanent record of any material typed upon it not only in printed form but also on punched paper tape. The latter can be subsequently re-used by feeding through a standard electromechanical tape reader connected to the main computer. Unfortunately these readers are little, if any, faster than a competent typist at the original teletyper; they have, however, the advantage of considerably greater accuracy.

By sensing the previously punched tape by electronic or photo-electric means it is possible to increase the reading speed from the customary 7 decimal digits per second to approximately 250 per second, the limitation being, not the sensing device, but the handling device for starting and stopping the tape. Effectively this thirty-five-fold increase in speed means that, assuming the main computer is adequately fast, 35 independent problem setters operating 35 separate tape producers can be kept busy.

Using punched card equipment, problem preparation time remains approximately the same, but by optimum use of the available card surface it is possible to input up to 300 decimal digits per second without the complexity of electronic or photo-electric sensing.

Before proceeding to more sophisticated input devices, it is appropriate to consider the means of operating the mechanisms described above in conjunction with a high speed computer. The important principle is that the computer has an operating speed several orders of magnitude greater than that of the input devices.

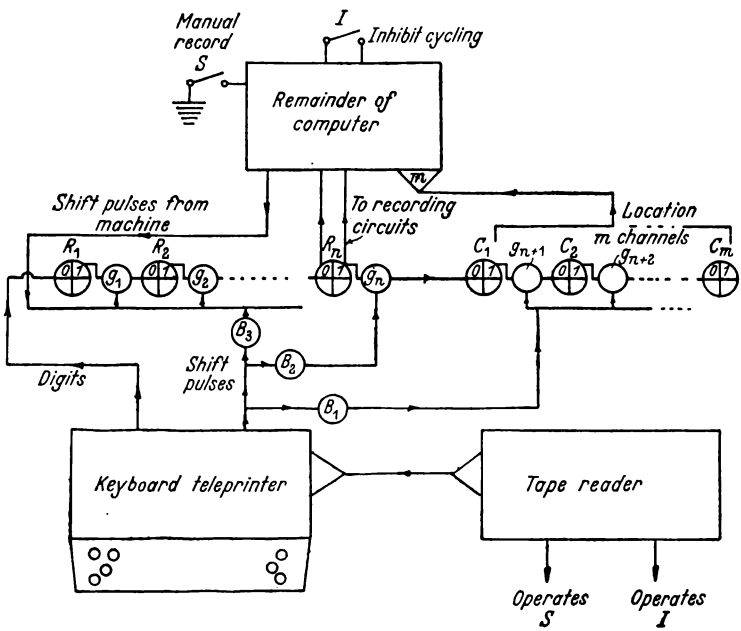


Figure 8.1.

This implies that for efficient and reliable operation, during input the computer should be the slave of the mechanism and not *vice versa*. *Figure 8.1* shows a typical means of operating a computer from a teleprinter. The computer is assumed to be of the serial operation type. When using the input device I is in the open position so that the machine is incapable of internal cycling. Data to be inserted is typed upon the teleprinter keyboard, the printer mechanism prepares a typed record and at the same time emits a set of serial impulses corresponding to the binary coded form of the number or symbol being typed. These impulses are applied to the input of a register R and at appropriate times the teleprinter mechanism supplies shift pulses to the register *via* buffer diodes B_1, B_2, B_3 and gates $g_1 \ldots g_m$.

Since the input number or word is destined to be recorded in the memory of the computer, the location must be specified. This is typed first, followed by the actual number, or word. Register R is connected to the control register of the machine, C, so that as the word digits and shifts arrive the location is eventually shifted out of R into the relevant positions in C. The important principle is that R and C can perform a shift operation in about 1 μsec whereas the digit input pulse is available from the teleprinter for a time of the order of 2×10^4 μsec; thus a large time exists for correct setting of the first register stage from the printer.

At this point the word and location digits can be checked by examination of the typed record (teleprinters often make mistakes, but rarely, if ever, generate a *wrong* electrical signal and type the *correct* number). S is then depressed and this causes the main computer to execute a record order which sends the word in R to the memory location indicated in C, suitable machine clock pulses being supplied to $g_1 \ldots g_{n-1}$.

This recording process is repeated until all information has been inserted, the machine cycle is then started by closing I.

To incorporate a standard teletype tape reader as input the same arrangement is used except that the printer is fed, *via* the standard connections, from the reader. The latter is so arranged that whilst tape is still to be fed a control brush keeps I in the open position, and a further control contact actuates S at the end of each set of rows of tape perforations. In this way the tape is fed to the computer and at the same time a printed record is produced which can be used as a check on programme accuracy in case of malfunction of the machine.

The use of a punched card reading device as computer input is even simpler. Assuming that *binary* data is to be inserted, 12 rows of 80 binary digits can be read from a single card. The reading is parallel on all 80 columns simultaneously so that direct connection to the individual stages of registers R and C can be made. The duration of digital impulse from each card row is usually greater than 20 msec so that ample time exists to record the required word in the specified position before the arrival of the next row under the sensing brushes.

The use of the punched card as a two-dimensional binary store is, however, somewhat unorthodox, and should the normal method of *decimal* representation shown in *Figure 8.2* be used, a slightly different input procedure is necessary.

Decimal numbers are represented by a punching as shown. Internal means exist, in punched card machines, for translating

these decimal digits into their binary equivalents and these translations can be fed, in parallel, to the *R* and *C* registers as before. It should be noted that this mode of use of a punched card is considerably less efficient and about 10 times as slow as the binary two-dimensional technique previously described.

Up to this point only one of the possible uses of an input device has been mentioned, namely its function as an initial supplier of data to the high speed store. In this respect both punched tape

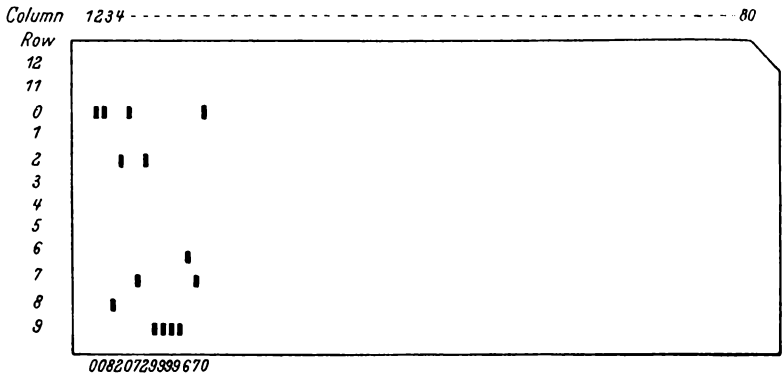


Figure 8.2. Standard 'Hollerith' punched card

and punched cards have adequate speed to match the operating times of current calculators. The second aspect of an input device is its availability as an auxiliary store of effectively infinite capacity. Here the machine may require the input to do two things, first to supply fresh programmes or numerical data when it has exhausted that already contained in its high speed store, and secondly to act as a repository for extensive tables of standard functions. Considering the first requirement, with a typical machine having an internal storage capacity of 512, 9 decimal (32 binary) digit words the filling time from punched cards need not be more than 30 sec, this is about the same as the filling time from punched tape photo-electrically read and is adequately fast for most purposes. On the other hand, punched tape, read by the standard teletype apparatus, would require about 12 min for the same operation, and this is definitely too slow.

The second application, that of hunting for an entry in a mathematical table, is more enlightening. Consider a machine repertoire of 10 tables, each of 1,000 entries, of 9 decimal accuracy. If this were randomly available to the machine on punched cards,

optimally used, an average hunting time of about 3 min would be required to look up a given tabular value, the same sort of time would also be required by a photo-electric punched tape reader. This time is about two orders of magnitude too great, even for applications requiring relatively infrequent use of tables. The equivalent time, on a mechanical reader, is over two hours.

This simple example shows the limitations of existing equipment when used as input devices to a high speed computer, and the following discussion will reveal the improvements to be expected from the application of new techniques to the problem.

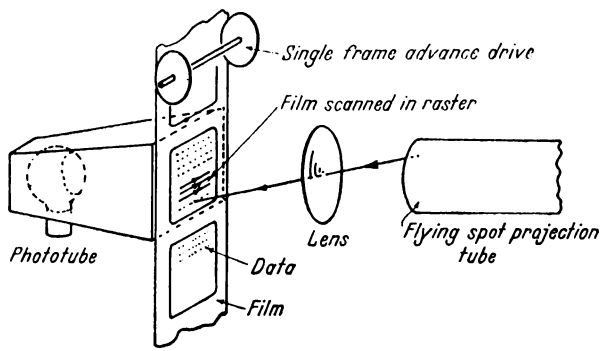


Figure 8.3. Schematic of film reader

Considering, first, the use of microfilm sensed photo-electrically, it is at once evident that, on account of the development time, such a medium is unlikely to be useful as a once only data input. On the other hand, for storing standard programmes and tabulated functions the situation is more hopeful. At a conservative estimate fifty 9 decimal digit numbers could be stored on 1 in. of 35 mm film; a conservative speed of running would be 25 in./sec which gives a filling time for a 512 word memory of $\frac{1}{2}$ sec, which is more than adequate for any existing machine. For the table hunting problem the average search time would be 4 sec, again very satisfactory. It is perhaps only fair to say that such a device does not exist, commercially at any rate, at the time of writing. A tentative method for the construction of such a reading device is shown in *Figure 8.3.* The film is advanced, one frame at a time, using a standard cinematograph projector mechanism. When any frame is in position a trigger pulse is emitted, by the mechanism, to the raster generator associated with a flying spot projection tube. The spot of light so produced is brought to a focus on the surface of the

film, and, in passing through the latter, is intensity modulated by the data recorded thereon. This intensity modulation is detected by means of the phototube shown on the left of the diagram and, after suitable shaping operations in the associated circuitry, is fed as serial digital information to the registers of the computing machine. This description is, of course, very incomplete but the engineering problems involved, such as registration of the film with precision ·005 in. and waveform operation at 50–100 kc/s, are well within the range of modern technical resources.

The next group of input devices to be considered is that based on the binary storage of data on ferromagnetic tape, wire or other coated surfaces. Since the use of such phenomena as memory devices is to receive detailed consideration in Chapter 12, only the relevant packing and speed data will be given here and the detailed description left until later.

Considering first the magnetic wire—basis of wire sound recording equipment—up to 100 binary digits per inch can be stored and the wire operated at speeds of up to 50 ft./sec. This means that, very conservatively, 600 words could be placed in the high speed store in 1 sec. This time is not quite the equal of that attained by the photographic method but equipment of this type already exists, and at the speeds and data packing densities quoted there are no engineering problems, either mechanical or electronic.

Magnetic tape can also be used for data storage. It has the advantage of allowing, on currently available tapes, up to 5 channels with a packing of 100 binary digits per inch and a speed of 5 ft./sec. This gives about the same filling speed as the magnetic wire and there seems no technical reason, except excessive read/record head wear, why this speed should not be multiplied by 5. The wear problem might be solved by the use of ferritic materials for the heads, a technique which gives other advantages and which will be discussed in Chapter 12.

The use of slow speed drums coated with magnetic material can give a filling speed for 512 words of better than $\frac{1}{4}$ sec; however, if such a device is available it appears much more profitable to use it as a part of the main store.

The remaining input devices are considerably more remote from practical realization, although they seem worthy of discussion as having attractive features. The possibility of mechanical recognition of printed characters and of their translation into electrical impulse patterns has long been a subject of discussion among computing machine designers. A number of possible solutions exist of which one, possibly the simplest, will be outlined here.

The paper bearing the letters or figures to be read is mounted on a platten as shown in *Figure 8.4.* This is movable, under the action of servo motors, in two mutually perpendicular directions. A glass disc, opaque except for a set of transparent replicas of the characters to be recognized, rotates at high speed about an axis normal to its plane. A light source, behind the disc, illuminates the transparent

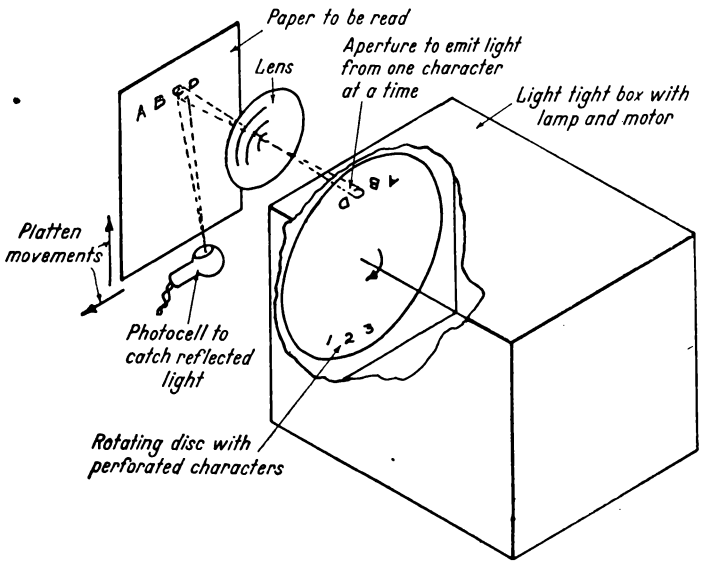

Paper to be read
Aperture to emit light from one character at a time
Lens
Light tight box with lamp and motor
A B C D
Platten movements
Photocell to catch reflected light
1 2 3
Rotating disc with perforated characters

Figure 8.4. Elementary character recognition device

regions in turn and the transmitted light is focused, by means of a lens, on to the surface containing the printed matter to be encoded. Assuming that the character concerned has been properly aligned with the lens system—an operation which can be performed automatically by means of the servo motors—only for one disc character will the light reflected by the platten be a minimum of pre-assigned smallness, this corresponds to the bright image of the disc character falling to exactly cover the printed character. When this minimum is reached, as detected by a suitably placed photocell, the angular position of the rotating shaft is made the basis of a device to transmit the coded version of the recognized character to the computer.

This simple device leaves many questions unanswered, for example the moving of the platten at an adequate speed and the need to obtain accurate size relationship between printed and image characters. These purely technical difficulties could, however, be overcome

in a variety of ways. Again, difficulty would arise from the fact that, for example a printed *O* would give a minimum with the image *C*; this can be resolved by operating the device in such a manner that *O* always precedes *C* in scanning sequence and that recognition inhibits the remainder of that particular cycle. As far as speed is concerned it would appear that such an apparatus might work at 50 characters per second and in this respect it would be superior to punched tape but inferior to most of the other input devices. One considerable advantage might lie in the fact that problems could be prepared for machine use on ordinary office typewriters which, in the case of a large number of coding operatives, would effect a considerable saving in the cost of teleprinter equipment.

The above simple device could be improved in numerous ways, for example, the flying spot technique can again be used to replace the rotating disc and obviate the necessity of a moving paper platten with consequent increase in speed. An advantage of this might lie in the fact that the characters traced by the flying spot could be directly generated from sample characters supplied with the problem sheet thus avoiding the need for standardized typewriters and even opening the possibility of recognizing handwritten script.

Yet another possible character recognition device might be based on what may be called the 'topology' of alphabetic and numerical characters, this would depend on the number of times suitably placed scanning lines intersected the printed character. Several such devices have been suggested but none seems so simple in principle as the covering scheme which has been described above.

As a final possibility for input devices it is proposed to discuss briefly the possibility of voice operated input equipment. It is not to be thought that this is a serious possibility for *computational input*, but in view of the probable use of computing machines for translation from one language into another, it is worth a short discussion.

Here again, possibly the simplest technique, although by no means the most efficient, would be to prepare a magnetic recording using an input clipping device to remove amplitude variation of the recorded word, and then to compare this record with a master recording containing all the words within the machine's vocabulary. When a complete match had been found the digitally coded equivalent of the vocabulary word would be read out into the body of the machine. There it could be operated upon, in conjunction with other input words needed to give 'context,' and eventually give rise to appropriate signals which would stimulate an output device, again

giving reproduction of a previously recorded sound, this time the translation of the input. The possibility of such a scheme rests upon the observation that amplitude standardization of recorded voice frequencies does not impare intelligibility.

Several difficulties exist in the above naïve discussion; in the first place it has been assumed that a given word is always spoken with the same pitch. It is likely that this might be overcome by replacing overlapping wave-form cancellation of the type suggested above, by harmonic analysis of the group of frequencies contained in the input word and the reduction of this spectrum to a series in terms of its fundamental. This would lead to a simple completely numerical process for the selection of the dictionary translation word and would remove the second major objection to the original scheme, that of designing an adequately fast store and comparison mechanism for the vocabulary words.

OUTPUT DEVICES

The first type of output device to be considered is a visual display by lights or neons. From the purely technical standpoint neon lamps are greatly to be preferred to those of the filament variety on account of their small size, speed of reaction and low power consumption. As a single output mechanism for a high speed calculator no form of luminous device can be considered adequate, however, since a human operator would be required to transcribe the displayed numbers on to paper. On the other hand, when a high speed computer is in use as an essentially special purpose machine, such display may suffice. For example, if the machine is being used to test the primeness or otherwise of input numbers displaying, in the composite case, the lowest factor, no more sophisticated device is needed. Such problems as this are, however, not the chief *raison d'être* of a large general purpose computer, and neons are many orders of magnitude too slow for general use.

It is convenient to consider here device (5) of our initial list. It is easily possible to photograph, on moving film, the numbers displayed on a neon bank. Using sensitive film and reasonably intense neons, an output speed of at least 50 numbers of as many digits as are required per second could be attained.

Teleprinters and teleprinter tape punches are the next form of output to be considered and these have been the most commonly used in current machines. The fundamental speed of these devices is approximately 7 decimal digits per second. This is at least one order of magnitude too slow for most calculating machines, for

example a typical slow machine could generate values of a 3rd degree polynomial for equal intervals of the argument at the rate of 50 values per second.

It is worth dealing here with a point in connection with the use of teletype apparatus as an output for a high speed computer. Normal teletype equipment is designed to accept a serial input consisting of 5 signals of 20 msec duration together with suitable controls. In the past many equipments have been modified to accept a 5 channel parallel input, the argument being that since 5 times as long a period is now available for setting up the solenoids, more reliable operation will be secured. Unfortunately this assumes that the solenoids which replace the original teletype magnet are of equivalent speed and tests have shown that this is not the case. For example, a typical 'Creed' teletype magnet showed an operating time, including subsidence of bounce of 2 msec, whereas a solenoid of the type used in most laboratories in the parallel input device required 100 msec for reliable operation. It follows that the safety factor for the teletype magnet is 10:1 whereas that for the parallel replacement is only 10:7. These tests led to an investigation of the causes of inaccurate operation of standard teleprinters when used as output devices for high speed computers, and the result emerged that nearly all of the trouble arose from the imperfect synchronization of the machine output to the teleprinter. Perfectly reliable operation was at once obtained when the teleprinter was modified so as to call for input from the main computer at the time required by its own cycle. This was simply achieved by fitting a form of commutator to the main teleprinter cam drive.

A more satisfactory output device is the parallel channel printer of standard punched card equipment. This will output up to 80 decimal digits at a speed of 100 operations per minute. By suitable programme arrangements the 80 channel parallel output can be split into 8 or 9 digit groups and an effective speed-up of the output by this factor obtained. A very considerable advantage of this type of output lies in the fact that it was designed with the requirements of numerical work in view, specifically to be extremely accurate and reliable.

The devices so far described may be called *primary* in the sense that they generate outputs directly interpretable by human beings. The next group of apparatus provides high speed output but in a form not suitable for ocular examination, the point being that such output can then be distributed among several 'interpreting' devices working at more moderate speed whilst the complex and expensive main computer is free for other work.

First of the high speed output devices is the magnetic wire, tape or slow drum. This will be familiar from its previous inclusion amongst the input devices; the important factor when output is considered is the possibility of its operation at the same digital repetition rate as that used in some computers—up to 100 kc/sec. This means, in principle, that no slowing up of the main machine is necessary on account of output requirements. As an example of this flexibility, a magnetic tape has been used to record all of the numbers held in the registers of a computer during the whole of a computation, a most valuable diagnostic tool. A minor, but real, advantage of magnetic devices lies in their erasability, which means that the large quantity of medium used in a high speed diagnostic device can be re-used as many times as required.

The use of film produced from a neon display has already been considered at the beginning of this section; a disadvantage of this form of output lies in the fact that, in the simple form at least, the photographic record will contain only dot imprints whose positions give the digital value of the actual output—at least in any system of notation other than binary. A more sophisticated device is the 'numeroscope,' this consists of a cathode ray tube and associated circuits so arranged that upon receipt of a binary coded decimal input, the actual decimal number is traced upon the screen. Arrangements are usually included to enable adequate page layout to be generated and the device is capable of extremely rapid operation. The only speed limitation is the stepping mechanism for the film, and if the required operational speed is high enough this can be obviated by moving the film continuously and using the switching properties of the cathode ray tube circuits to generate the numerical pattern for photography only at those times needed for the page layout. Even without such complexity it should be possible to construct a stepping mechanism with a speed of 100 impulses per second and this can be considered adequately fast for most purposes.

Finally, as with the input devices, it is worth considering the possibility of generating a sonic output from a computing machine, not with a view to its use in strictly mathematical applications, but rather for the possible application of computers as translating or interpreting devices. The production of musical sounds from a high speed calculator is trivial, all that is needed is an audio amplifier and speaker connected to the output of a register or to the main store. If a single unit is stored in the register and is caused, by means of a suitable iterative routine, to cycle with a given frequency, a note of corresponding pitch will result at the speaker. Harmonics

of the fundamental can be produced by the simple process of altering the disposition of the number of units in the register thus:

Fundamental	1111111100000000
1st Harmonic	1111000011110000
2nd Harmonic	1100110011001100
3rd Harmonic	1010101010101010

and other non-integral ratios can be similarly obtained to give the notes of the common chord. This application of a computer has already been realized and made the basis of an adequate performance of 'God save the Queen.'

The production of the spoken word requires a more elaborate equipment than that needed for the generation of music. Two schemes are available; the first is the naïve one of recording each word to be spoken on the magnetic medium and arranging for a suitable coded output from the machine to select that portion of the record containing the word to be 'spoken' and to gate it to the audio output device. This scheme, although it might give very realistic output, would be complicated to engineer and would involve an exceedingly bulky apparatus. A simpler technique would be to use the method of speech output developed for the 'Vocoder' or 'Voder.' Essentially this generates the relevant speech sounds by the sequential combination of a very small number of basic tones which, in the present application, might conveniently be recorded on magnetic tapes and be selected by the machine as required.

RETROSPECT

Looking back over the preceding discussion of input/output devices it is possible to make some general observations concerning those systems which appear best at the present time.

In the first place, any calculator should be fitted with a manual input and a visual output of the simplest and most reliable kind as an aid to the service engineer who has to maintain it. All of the more sophisticated terminal organs require that a large portion of the main machine be operative before they can be used. When a breakdown occurs this is unlikely to obtain and in the absence of a simple device the unfortunate engineer is reduced to the laborious process of input *via* a suitably connected screwdriver and connector, and output *via* a valve voltmeter! The switches used should be of reliable construction, such as those used by the Post Office for telephone switchboards; the neon lamps should have

stable striking characteristics and small dissipation, the type N.T.2 having proved satisfactory.

As far as the automatic and high speed input/output are concerned the choice depends largely upon the amount of use required of the main machine. In small installations the combination keyboard teleprinter, reperforator and reader is probably adequate, especially if the internal store of the machine is of the magnetic type so that working instructions do not have to be input after each shut-down. If the machine is of the ultra fast variety, and is to be kept fully operational, there seems little doubt that the flexibility and simplicity of the system (several teleprinters) (magnetic wire or tape) is the best. This is especially true where the internal storage of the machine is of the type which requires refilling after shut-down, or if the work of the machine is to involve considerable access to previously computed mathematical tables.

9

GATES

In discussing the specific circuits and devices used in the construction of a high speed automatic computing machine it is a matter of some difficulty to find a logical starting point. If, as might seem appropriate, the memory or store were taken first it would at once be found that the organization of this required the use of a number of interconnecting elements whose properties and function had not previously been studied. The same argument holds with even more force in the case of arithmetic unit components which are constructed entirely from simpler and more fundamental sub-units.

Careful consideration suggests that the proper element to receive consideration should be either the single digit storage element or the gate. The latter has, in fact, been chosen because the physical form of a gating element is such that it can be applied in a machine using almost any high speed storage device. The design of the single digit store, on the other hand, depends to a large extent on the particular form of the main store and particularly upon its speed of operation.

THE ELECTROMAGNETIC RELAY

The simplest and most versatile gate element, although also the slowest, is the electromagnetic relay. It consists essentially of a soft iron armature *a* (*Figure 9.1* (*a*)) normally held out of contact with a core *c* by means of a spring *s*. Upon application of an electric current to the leads, *AA*, of a coil of wire wrapped about *c* the armature *a* is attracted against the force of *s* to make contact with *c*. The current to be 'gated' is supplied to *a via* a lead *B* and is then directed to one of two outputs *C* and *D* according to the presence or absence of current on the leads *AA*.

For convenience of drawing, a relay is usually represented as shown in *Figure 9.1* (*b*). The coil is replaced by the rectangle and the contacts by the structure *B*, *C*, *D*.

Relays are available with several coils and a variety of contact arrangements, but in general it is true to say that the greater the number of contacts, the slower the operating time of the device.

A typical example of a fast relay is the Siemens high speed relay which has up to two operating coils and a single change-over contact; this has an operating time on make or break of about 1 msec. The *British Tabulating Machine Company* has produced a similar relay with up to 8 change-over contacts and an operating speed of only

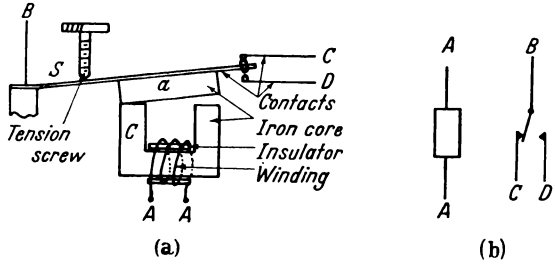


Figure 9.1 Electromagnetic relay

1·5 msec. Typical of a slower relay is the P.O. 3,000 type. This can have up to 3 coils and up to 6 change-over contacts, its speed of operation is about 20 msec. Current handling capacity of the Siemens type relay is up to .2 A, and that of the P.O. 3,000 type, with suitable contacts, up to 5 A.

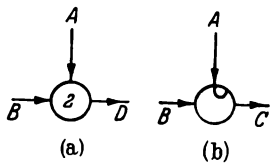


Figure 9.2 Logical equivalents of relay

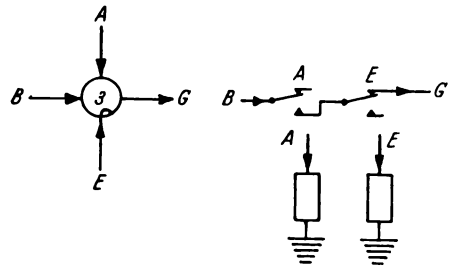


Figure 9.3 Use of relays as multiple gates

It is seen that a single relay is essentially a 2 gate of the type used in previous chapters. *Figure 9.2 (a)* shows the logical notation lettered for comparison with *Figure 9.1 (b)*. By rearranging the contact connections in accord with *Figure 9.2 (b)* an inhibitory or 'not' gate is produced.

By the use of relays in series it is possible to produce multiple gates and multiple gates with inhibitory connections. A typical example of this application is shown in *Figure 9.3.* Here, the coincidence of

inputs on leads *A* and *B* leads to an output on *G* so long as *no* input is present on *E*.

In logical terminology a gate of the type shown in *Figure 9.2* (*a*) is known as an 'and' gate, and that shown in *Figure 9.2* (*b*) as a 'not' gate; the third conventional logical element is usually called an 'or' gate. This can be realized in a number of ways by means of relays, one of these being shown in *Figure 9.4*.

The relay circuit has been drawn so as to bring out the close analogy with the basic logical diagram used, for example, in the half adder of Chapter 6.

The close analogy of relay circuitry with the elements of symbolic logic has been made the basis of a number of applications of that calculus to the design of such circuits. Boolean algebra, as it is

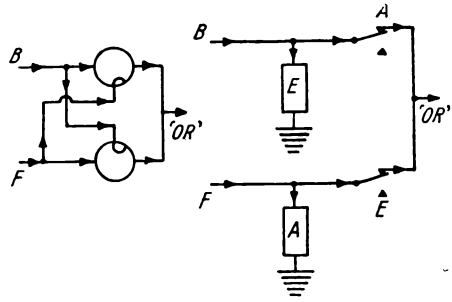


Figure 9.4 Use of relays as 'or' elements

called, is a convenient method, in principle at least, for the solution of design problems; in practice, however, it is almost invariably quicker to proceed by more intuitive means. It is almost a truism to say that some engineers will design an optimum relay circuit in a very small time without any reference or knowledge of propositional calculus, whereas others, less gifted, will never design a good circuit using any means.

Before leaving the subject of relay gates it is appropriate to make a few remarks on practical matters. For general computer work the best form of high speed relay seems to be that having a coil resistance of 2,000 Ω, this operates very reliably and fast when placed in the anode circuit of a pentode type valve capable of giving a continuous current of 15–20 mA. For reliable circuit operation a voltage in excess of 50 should exist across any relay contact just before closure, this has been shown to produce a microscopic weld on contact which ensures negligible contact resistance. The normal contact

material for high speed relays is platinum; it appears that rhodium would be more suitable from the contact resistance viewpoint, but the latter material does not seem to have been used. Contact pressure in the released position should be at least 17 g and on 'make' slightly greater than this.

For relays of the P.O. 3,000 type a coil resistance of 2,000 Ω again seems optimum for valve drive circuits. Contacts are generally of silver and the pressure depends almost entirely upon the number of sets mounted upon the relay.

THE DIODE OR RECTIFIER

The relay, by virtue of its mechanical construction, is essentially a slow device, and in order to produce a faster gating operation it is necessary to have recourse to electronic techniques. The simplest form of electronic gating element is the diode or rectifier. This can exist in several distinct forms; in the first a suitably coated cathode is heated by an external source of current and emits electrons. At some distance from the cathode and mounted with it in an evacuated glass envelope is a collector plate or anode. When a positive potential is applied between anode and cathode electrons flow from the latter to the former giving rise to a current in the external circuit; a potential applied in the reverse direction, however, produces no corresponding electron transport and consequently no current flows.

An exactly analogous situation exists in certain devices constructed from the so-called semi-conductors. These possess the property, when placed between suitable metal surfaces, of being capable of transporting electrons from one metallic contact to the other for applied potentials of one sense, but of behaving as insulators for the reverse potential.

Diode gates are not the precise equivalent of the relay for a number of reasons. Apart from physical characteristics, which will be considered later, the diode can be considered as only one half of a change-over relay contact system. Some typical examples of diode gating circuits are shown in *Figure 9.5.*

In *Figure 9.5* (*a*) is shown the diode equivalent of the 2 gate. V_1 and V_2 are the two diodes required for efficient operation of the circuit, the potential at B is set by the resistor R_B connected to a source of, say, + 10 V potential. Diode V_1 and its associated resistor R_A have a combined resistance small, say $\frac{1}{10}$, compared with that of R_B, so that in the absence of an input on line A the potential at B is approximately 1 V. R_D is a bias resistor and maintains the

cathode of diode V_2 at a potential of +21 V, that is 20 V positive with respect to its anode. Under these conditions a +20 V pulse applied at B will drive the anode of V_2 to a potential of +20 V which is just insufficient to produce conduction and thus output at D. If, on the other hand, A is in receipt of an input voltage of +30 V, say, V_1 is normally cut off and the potential at B is +10 V. Under these circumstances an input pulse of +20 V at B will produce an output pulse of +11 V, through the diode V_2, at D.

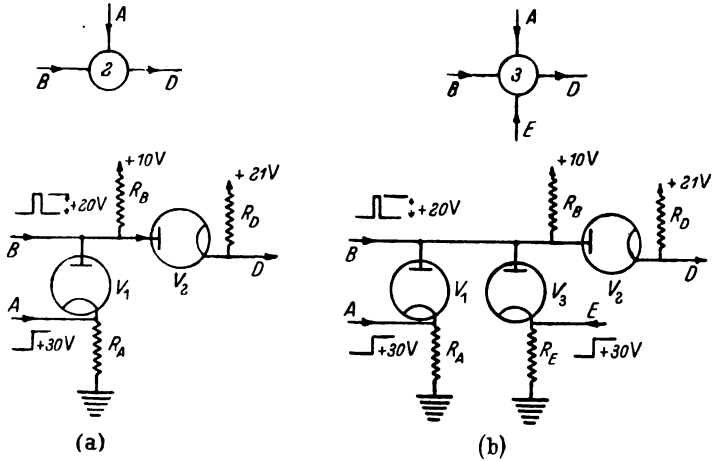


Figure 9.5 Typical diode gating circuits

A more complex variant of this circuit is the 3 gate shown in *Figure 9.5* (*b*). This is identical with *Figure 9.5* (*a*) except for the addition of an extra gate input E at V_3. The operation is the same as before except that now, since both R_A and R_E are much smaller than R_B, the potential at B remains approximately +1 V unless an input occurs on both A and E simultaneously.

To produce an inhibitory input it is merely necessary to arrange that the input voltage at A (*Figure 9.5* (*a*)) is normally +30 V, in which case the gate is open, and that a negative gate voltage is applied when inhibition is required.

This type of diode gate is capable of great speed and good long-term stability when properly constructed; an essential feature is that the potential at B is set by the gate voltage at A—a process which can be carried out relatively slowly—and the operating voltage pulse can then pass diode V_2 at high speed and with good rise time of leading and trailing edges. An example of a diode

gate in which this response is not so good and in which, also, a certain amount of 'break through' occurs on application of the gating voltage is shown in *Figure 9.6.* In this circuit, in the absence of inputs at A and B, if $R_D = \frac{1}{10}(R_A + V_1) = \frac{1}{10}(R_B + V_2)$ the standing voltage at D is approximately +1 V. If either A or B receives a positive input of more than +2 V the potential at D rises to approximately +2 V; if, however, both A and B receive simultaneous inputs of +20 V or more, the potential at D rises to +20 V. The same arrangement applies equally to any desired number of inputs.

The difficulty with this type of circuit lies in the fact that to ensure a fast rise time of the output pulse the time constant of R_D and the output circuitry must be low. For example, if the output capacity is 20 $\mu\mu$F and a $\frac{1}{25}$ μsec output rise time is desired, the value of R_D

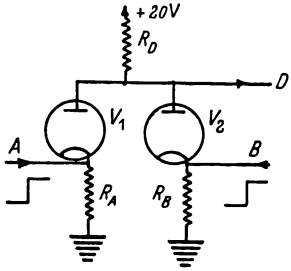


Figure 9.6 Simplified diode gate

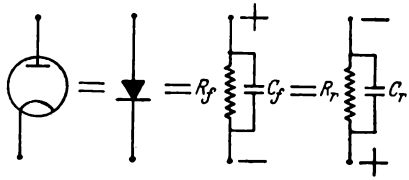


Figure 9.7 Equivalent diode circuits

is easily shown to be 2,000 Ω; this means that the total resistance $(R_A + V_1)$ must be of the order of 200 Ω for satisfactory operation and, since this is the resistance of normal diodes, the resistance R_A has to be zero. This shows the inherent limitation of the simplified circuit when required for the generation of very sharp output pulses.

It is now possible to assess the pros and cons of the diode gate. In its favour are the facts of its extreme simplicity, its constructibility from semi-conducting elements, and its long term stability. On the debit side are the facts that, in general, the output pulse is of less magnitude than any input pulse, making impossible the interconnection of several such gates without booster amplifiers; and also the gating inputs (i.e. A, B, E, *etc.*) are of very low impedance (electronically speaking at least) making it necessary to drive them from a power amplifying device of the cathode follower type.

To conclude this short account of diode gating circuits it is

appropriate to give a summary of the physical characteristics of some typical elements.

In *Figure 9.7* are shown various representations of the diode element. From the left, the first symbol represents the valve, or thermionic diode; the second the semiconductor diode, selenium or germanium; the third the equivalent circuit in the forward (i.e. conducting) direction; and the fourth the equivalent circuit in the reverse direction.

In *Table 9.1* the characteristic values of the constants for various types of diode are given. The quantity V_r(max) is the greatest

Table 9.1

Type	Construction	R_f	C_f	R_r	C_r	V_r(max)
6AL5	Thermionic	200Ω	5μμF	100MΩ	5μμF	> 300 V
1N38	Germanium	200Ω	2μμF	100KΩ	2μμF	<120 V
M.R.1	Selenium	6,000Ω	22μμF	20MΩ	22μμF	< 50 V

voltage which may be safely applied in the reverse direction without destruction of the element. Figures given refer to typical commercially available types, it is reported that germanium diodes have been produced with forward resistances of only $\frac{1}{10}$ the value of those quoted. Selenium diodes can be had in a range of resistances, the important factor being that

$$(\text{Forward Resistance} \times \text{Capacity}) \simeq 10^5\ \Omega.\mu\mu\text{F}$$

or alternatively that their time constant is approximately $\frac{1}{10}$ μsec.

Experience has shown that selenium diodes form adequate gates at frequencies up to 40 kc/s, whilst germanium and thermionic equivalents are satisfactory up to 2 Mc/s.

The triode valve is in some sense more akin to the electromagnetic relay when considered as a gating element than is the diode. Particularly is this the case in the isolation which it affords between input and output. Without going into detail the triode may be considered as a diode into which has been inserted a control element. This element, known as the grid, is usually in the form of a wire mesh interposed between cathode and plate. When its potential approximates that of the cathode a positive potential applied

between plate and cathode is still able to draw electrons through the interstices of the mesh. When, however, the grid potential is sufficiently negative with respect to that of the cathode, all electrons are repelled back to the latter and none pass to the anode. In terms of relay circuitry the triode value may be represented by the circuit of *Figure 9.8.*

It is to be noted that a diode element has been inserted in series with the relay contacts, this is to conform with the physical fact that

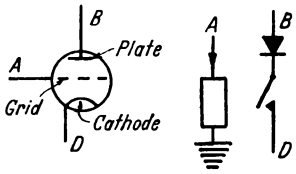


Figure 9.8 Triode and relay/rectifier equivalent

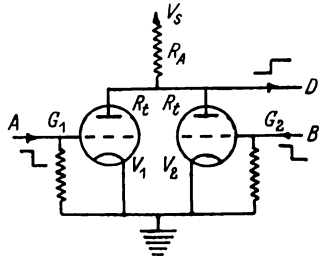


Figure 9.9 Anode coupled triode gate

the plate/cathode circuit of a triode has the properties of a diode and will not pass current in the reverse direction. Apart from its control function an essential property of the triode is that the grid circuit can control large currents in the plate/cathode circuit without itself drawing appreciable current.

The first application of triode valves to the construction of a 2 gate is shown in *Figure 9.9.* This circuit is the equivalent of the diode circuit of *Figure 9.6,* when no signals are applied at inputs A and B both triodes behave as the diodes of the latter figure and maintain the potential at D equal to:

$$\tfrac{1}{2}R_t/R_A + \tfrac{1}{2}R_t) \times V_S$$

If $R_A \simeq 10\ R_t$ this means that the normal voltage at D is about $\frac{1}{20}$ of the supply voltage V_s. When either of the triodes is cut off by a negative potential at its grid its resistance becomes effectively infinite and the voltage at D rises to $\frac{1}{10}$ of the supply voltage. When both triodes are cut off, however, the voltage at D rises to the full supply voltage V_s.

Compared with the diode circuit this version has the advantage of very high impedence input at A and B, but the disadvantage that since the internal resistance of the triodes R_t is usually about 7,000 Ω, an anode resistance R_A of 70,000 Ω must be used and an

output waveform rise time—assuming 20 μμF capacity—of about 2 μsec is the best that can be hoped for.

To produce an *n* gate it is merely necessary to connect additional triode elements to the common anode point remembering, however, that each additional valve introduces an output capacity of at least 10 μμF with consequent additional variation of the output rise time. It must also be remembered that *small* output variations will result from any change in the input conditions and it is generally necessary to follow this type of gate with a clipping stage.

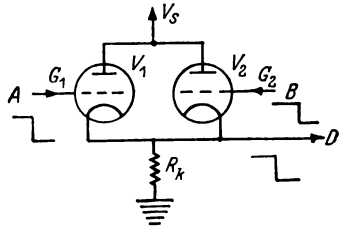


Figure 9.10 Cathode coupled double triode gate

Next in the list of possible triode gate circuits is that shown in *Figure 9.10.*

Here use is made of the fact that if the cathodes of a pair of triodes are connected together and taken to earth *via* a resistor R_K, then if the grid potential of *either* stage is V_g the cathode potential is approximately the same when R_K is suitably chosen. The effect of this is that if *both* grids G_1 and G_2 are kept at a common potential a negative pulse applied to *either* of them separately will produce little voltage change at *D*. When *both* receive simultaneous negative pulses, however, the potential at *D* falls to whichever grid voltage is the greater.

Exactly as in the anode coupled gate a number of such devices can be connected together to form an *n* gate.

Using normal valves a value of R_K of about 20,000 Ω can be used with a supply voltage $V_s = 150$ V, this means that the 'rise' time attainable with the device is somewhat better than that in the anode coupled case. It has been suggested that the 'rise' time should be that of the valves themselves, that is, equivalent to that obtainable across 500–1,000 Ω. This is not so, since for *negative* outputs the output impedance of any cathode follower is equal to that of its cathode load R_K.

A very considerable advantage of the cathode coupled gate circuit lies in the fact that the output potential is approximately equal to

the input potentials; this means that stages can be series connected without the elaborate biasing networks required in the anode coupled gate.

Disadvantages are, first that the cathodes of the triodes may, if indirectly heated, be operating at a considerable positive potential with respect to the heaters and care must be taken to observe the manufacturers' limits on this voltage; and secondly, that the negative output resulting from the gate has to be clipped to remove the breakthrough resulting from the operation of A or B separately. The latter clipping operation is complicated by the negative nature of the pulse and the want of an efficient clipping amplifier for such waveforms.

Many of the faults of both of the above circuits are removed in the combined version shown in *Figure 9.11.*

In this circuit the potential on grid B is normally low, say 60 V, that on grid A is high, say 90 V, so that the potential across R_K is effectively that at A, i.e. 90 V. If, now, A receives a negative impulse of 20 V amplitude, potential at R_K drops to 70 V which is

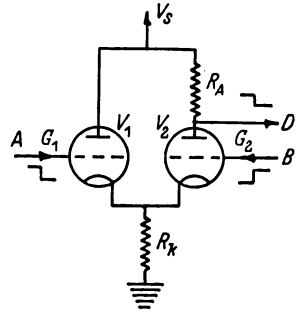


Figure 9.11 Anode output cathode coupled gate

still below the cut off voltage of V_2. In the same way, if A is at 90 V and B receives a 20 V positive impulse the voltage on G_2 rises only to 80 V so that V_2 is still cut off. If, however, A receives its negative pulse at the same time as B is driven positive, conduction through V_1 is transferred to V_2 with a resulting negative output on D.

The important features of this circuit are first that no breakthrough occurs on D for changes on A or B singly, and second, that since the device is essentially a current-switching one, the resistance R_A can be made as low as required with consequent improvement in output fall time. Typical values are $R_K = 22{,}000\ \Omega$, $R_A = 4{,}700\ \Omega$, which give an output rise time across 20 $\mu\mu$F of $\frac{1}{10}$ msec.

Just as in the other devices described a multiple gate may be formed by the addition of as many type *A* input stages as are required.

A considerable advantage of this gating system lies in the fact that it may be modified, by the substitution of a pulse transformer for R_A, so as to give a positive output waveform. This is a particular advantage if subsequent pulse amplification is required, since appropriate biasing for the amplifier stage is readily obtained as shown in *Figure 9.12.*

Before concluding the discussion of the triode as a gate element, it is perhaps worth mentioning a further mode of connection which has been applied, this is shown in *Figure 9.13.*

Here the triodes are series connected with a suitable anode resistor as shown. If *A* and *B* are held at a potential of, say, -10 V, both triode sections will be non-conducting and the potential at *D* will be equal to V_s. If either *A* or *B* is driven positively, current flow

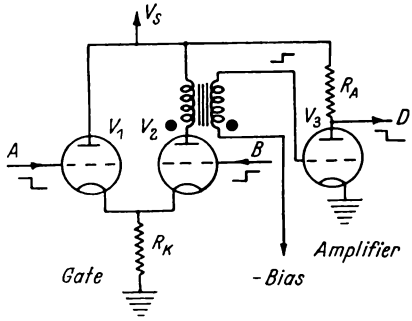


Figure 9.12 Modification for positive output to amplifier

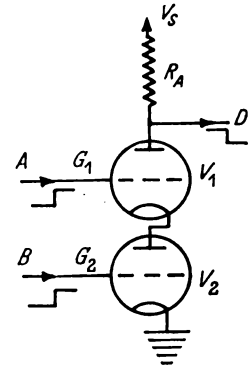


Figure 9.13 Series connections of triodes

will still be inhibited by the remaining triode; if, however, both *A* and *B* are simultaneously in receipt of positive inputs, both triodes conduct and a negative output appears at *D*. Unlike the previous circuits this one does not appear to have had extensive trial, but it would appear that it offers considerable promise for certain applications, as shown by its successful use in the *A.C.E.*

From a practical point of view the valve types 6J6, 6SN7 and 12AU7 or their equivalents are particularly suited to the triode gate circuits previously described; for details the reader is referred to the relevant manufacturers' data sheets.

MULTI-GRID VALVES

Just as it proved possible to control the electron flow in a diode by the insertion of a grid, so also can additional grids be inserted into the triode valve to give additional control function. At present only one such multi-grid valve, the 6AS6, seems to be suitable for use in computing machines. This has the unique advantage over other valves of multiple input type that the voltages required to effect cut off are the same on both grids; in the case of other valves this is not true and in some instances the second grid G_2 in *Figure 9.14* requires 5 to 10 times the applied voltage as G_1.

The operation of this device as a gate is precisely the same as that for the series arrangement of triodes previously described in connection with *Figure 9.13.* If the valve is held cut off by suitable bias

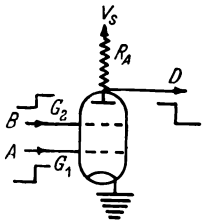


Figure 9.14 Multi-grid valve as gate

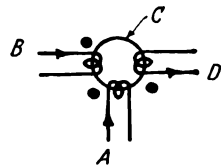


Figure 9.15 Schematic of magnetic gate

potentials applied at A and B, only by simultaneous application of positive gating voltages at these points will the valve conduct and in so doing produce a negative output at D.

Whilst in principle, at least, the multi-grid valve is an excellent gating device it suffers, in currently available types, from the disadvantage that the resistance to ground in series with G_2 must not exceed a few thousand ohms. This makes the device an essentially low input impedence one and severely limits its application.

MAGNETIC DEVICES

Before leaving the subject of gates it is necessary to consider the application of purely magnetic devices to the performance of switching operations. Whilst in principle almost any ferromagnetic material has sufficient hysteresis to make it suitable for gating operations, the use of such devices did not become really advantageous until materials with an almost exactly rectangular **B.H.**

loop were available. Several such materials are now obtainable commercially. In the U.S.A. *Deltamax* and in this country *H.C.R.* and *Permalloy F* are perhaps the best known.

To illustrate a simple gating operation performed with the aid of a magnetic element, consider the circuit of *Figure 9.15.*

A toroid of the magnetic material is furnished with three windings, *A*, *B* and *D*. If a current pulse, of sufficient magnitude, is applied to *A*, then the magnetic state of the core is taken to *l*, say, in *Figure 9.16.*

On removal of the pulse on *A* the magnetic state of the system returns to *r* in which practically the whole of the input energy remains stored in the core. If now a pulse is applied at *B*, in such a manner as to have the reverse magnetizing effect as that applied at *A*, the core undergoes a large flux change to *u* and relaxes to *t*,

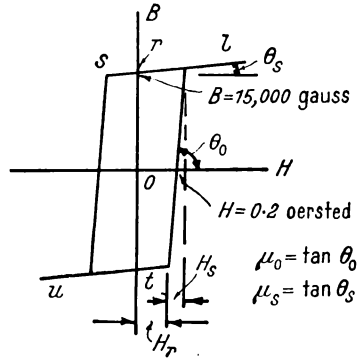


Figure 9.16 Magnetic characteristics of 'Deltamax', 'H.C.R.' and 'Permalloy F.'

and by normal transformer action an output pulse will have appeared at *D*.

If, however, the initial pulse on *A* had not been applied, the pulse on *B* would have found the core already in state *t* and the resulting change to *u*, and back, since it involves little or no change in **B** will produce only an insignificant output at *D*. It should be noted that a large pulse (of the reverse sign to that discussed) appears at *D* when *A* is primed; this can, if undesirable, be removed by means of a series or shunt diode suitably connected. Similarly the input at *B* may produce output, not only at *D* but also at *A* and this may require removal.

An advantage of the above gating system not possessed by the electronic devices previously described lies in the fact that the magnetic ring is also a single digit storage device, the priming pulse

on A does not have to be (and in fact must not be) coincident in time with that on B and this may be an advantage in certain applications.

Several variants of the above circuit exist; for example, A may normally be in receipt of a polarizing current which maintains the magnetic state of the core at l, pulses applied at B are insufficient to take the magnetic state beyond s and consequently produce no output. If the polarizing current on A is removed, or reduced, pulses at B cause change of magnetic state past s to u and consequently produce an output. Used in this manner the gate can be extended to several inputs; when this is done, however, it becomes effectively an analogue current adder producing output only above a certain level.

In order to make use of materials (specifically the ferrites) which

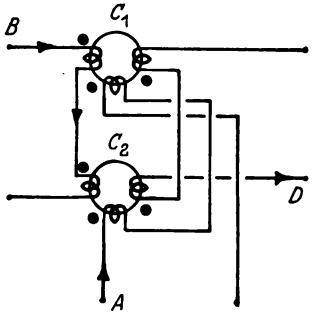


Figure 9.17 Differential gate

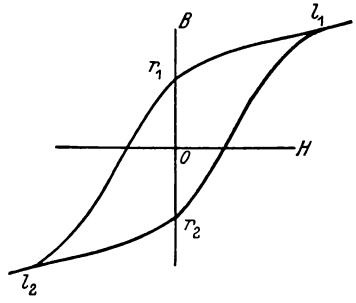


Figure 9.18 Magnetic states in differential gate

have otherwise desirable properties, but insufficiently rectangular hysteresis loops, the differential circuit of *Figure 9.17* was devised.

Two toroids C_1 and C_2, are used. Windings B are connected in the same sense in series, windings A and D, respectively, in series opposition. Pulses applied to B, in the absence of input on A, produce equal and opposite outputs on the two coils of D and thus no resultant output. When any pulse whatever is applied to A, cores C_1 and C_2 are magnetized in opposite senses. Suppose that in *Figure 9.18* C_1 is in state l_1 and C_2 in state l_2 and that on removal of the pulse on A the cores relax to r_1 and r_2 respectively.

When the B pulse is applied, C_1 changes in state from r_1 to l_1 and C_2 from r_2 to l_1; the resultant output on D, which is generated by the difference between these two flux changes is effectively that due to a flux change from r_2 to r_1 and, even in materials of poor remanence, is large.

Had the priming pulse on *A* been in the reverse sense to that considered above, the output at *D* would have been reversed, thus the differential gate has the additional property of forming a three-state storage device.

From the practical standpoint it may be mentioned that a suitable construction for the simple gate of *Figure 9.15* consists of two turns of ·001 in.–·008 in. H.C.R. ·375 in. dia. × ·125 in. high mounted in a protective box and toroidally wound with three sets of 200 turns of # 38 lumex wire.

For the differential gate suitable cores are ·125 in. × ·125 in. section × 1·00 in. dia. Ferroxcube 4 C with 200 turn windings; this structure has proved satisfactory in tests with ·25 μsec pulses.

10

SINGLE DIGIT STORAGE

IN THE design of control equipment for high speed calculating machines it is necessary, first to extract the instruction word from the main storage device, and then to staticize it in such a way that the individual digits are available for examination over a period much greater than the fundamental repetition interval of the machine. Furthermore, the gates of certain machine units, e.g. the adder, have, in serial operation at least, to be held open for the duration of the operation. These functions, as well as those typified by the storage of R_{n+1} in the multiplier circuitry, call for a unit which may be called a single digit store.

Many forms exist for this device, and in general the design is closely interlinked with that of the main high speed store; since, however, the design of the latter by no means fixes absolutely the form of single digit storage used, it seems worth while devoting a chapter to a fairly exhaustive survey of such devices.

THE ELECTROMAGNETIC RELAY

This is possibly the simplest example of single digit storage. The relay is connected as shown in *Figure 10.1.*

The normally off contact C has one side taken to a battery or other source of power, its make is taken to the relay coil R, the other end of R being returned to the battery. If coil R is energized by a current pulse of sufficient duration applied at A the relay will go into the on position, C closes and R is supplied with current independently of the input of A. In order to reset the device it is necessary to interrupt the current by means of an external switch.

A more elaborate variant of this circuit is shown in *Figure 10.2*; here the relay is provided with two independent coils R_1 and R_2 and several contacts, C_1, C_2, *etc.* Current applied to R_2 *via* A energizes the relay which then holds on contact C_1 and its other coil R_1. A resistor r placed between C_1 and R_1 enables the device to be cleared by closure of D, thus earthing R_1. The additional coil R_2 isolates the input from the hold and output currents, and contacts C_2 *etc.* provide independent output channels if required.

This type of store, in common with other relay devices, provides operate and release times from ·001 sec upwards, and in electronic computing machinary the chief application is in the interconnection of input/output apparatus to the electronic circuitry.

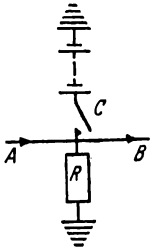


Figure 10.1 Simple relay store

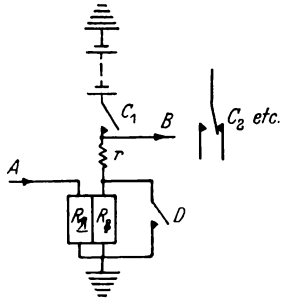


Figure 10.2 More elaborate relay store

THE GAS FILLED THYRATRON

The thyratron, or electronic relay, consists of a structure similar to that described for the triode valve; instead of being mounted in an evacuated enclosure, however, the electrodes are sealed into an envelope containing mercury vapour, neon, argon, *etc.* The effect of the presence of these heavy ions is to break up the electron cloud normally surrounding the cathode of a hard valve and thus to allow the passage through the valve of much heavier currents than are possible with vacuum valves. The second effect of the vapour is to render the valve insensitive to potential variations on its grid when it is in the conducting state. Such a valve is thus a two-state device and once conducting can only be brought to the non-conducting state by reduction of its anode potential to a low value.

Typical valves of this type, such as the 2D21, will pass currents of up to ·5 A, will take up the excited state in 20 μsec and require up to ·1 msec for restoration to the quiescent state.

In computing machines the chief use of thyratron-type valves is at the output side where the low level electronic devices of the machine are required to operate heavy print magnets and allied circuitry. Great care must be taken to avoid internal generation of spurious pulses by thyratrons as the valves are particularly liable to this fault and the radiated interference can produce serious malfunctioning of low level circuits.

ELECTRONIC FLIP-FLOPS

This type of circuit exists in two distinct forms, monostable and bistable. By these expressions it is meant that the circuits, when triggered into an excited state, either return after an interval to their original state without further external action, or remain in the excited state until triggered out of it.

A typical example of a monostable circuit is shown in *Figure 10.3.* Here, the grid of V_2 is taken to high potential *via* a resistor R large enough so that the permissible grid dissipation of the valve is not exceeded. Under these conditions current flows through R_{A_2}, V_2 and establishes a potential across the cathode resistor R_K. The grid of V_1 has its potential set by the network $R_G R_E$ so that, with R_K at the potential generated by the conduction of V_2, V_1 is cut off.

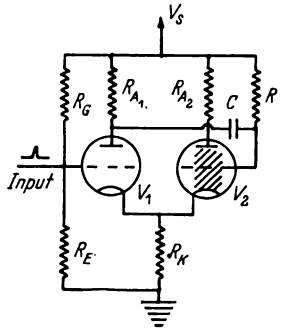


Figure 10.3 Mono-stable flip-flop

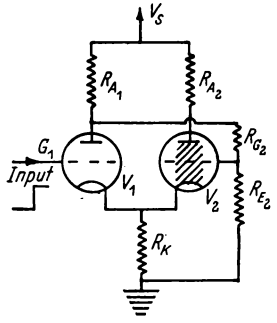


Figure 10.4 Schmidt trigger circuit

The anode of V_1 is connected to the power supply *via* R_{A_1} and to the grid of V_2 *via* the condenser C. The conditions described are stable.

If now the grid of V_1 receives a positive pulse of sufficient amplitude, V_1 will be rendered conducting, the potential at its anode will fall and the resultant negative pulse will pass *via* C to the grid of V_2. V_2 will tend to conduct less strongly, the potential across R_K will drop and the potential at the grid of V_1 will be increased. This cycle is regenerative and will cease only when V_1 is fully conducting and V_2 is cut off. At this point condenser C starts to charge *via* R, and when it reaches the point at which V_2 starts to conduct, the reverse of the above process occurs and the circuit restores itself to its initial state.

This type of circuit has been much used for the production of delays, the rising pulse at the anode of V_1 being delayed with respect

to the trigger pulse at its grid by an amount depending on the time constant *RC*.

A circuit intermediate between the monostable flip-flop described above and a bistable circuit is the Schmidt trigger shown in *Figure 10.4.*

In this circuit there is a resistive connection between R_{A_1} and the grid of V_2 *via* R_{G_2}, the grid of V_2 is also taken to ground *via* R_{E_2}.

For a suitable low d.c. voltage applied at the grid of V_1, V_2 is conducting and V_1 is held cut off by the voltage developed across R_K. As the potential of G_1 is increased these conditions are maintained until V_1 starts to conduct; at this point a regenerative cycle sets in and the circuit triggers into the state in which V_1 is conducting and V_2 is cut off. For suitably chosen component values this regeneration is extremely rapid and the circuit is often used to shape or sharpen a slowly varying incident waveform. A further advantage of the Schmidt circuit lies in the fact that, since G_1 is not connected to any other circuit element it is a very high impedence input and the device will give rapid triggering action when connected to such devices as selenium matrix function tables.

Suitable valves are the 6J6 and 6SN7, and suitable component values $R_{A_1} = R_{A_2} = 22$ KΩ. $R_{G_2} = 33$ KΩ. $R_{E_2} = 22$ KΩ. $R_K = 10$ KΩ and $V_A = 150$ V.

Since R_{A_2} is not loaded by a grid network as is the case for R_{A_1}, the potentials at the two anodes are not symmetrical; if such symmetry is required R_{A_2} may be returned to a supply voltage of +90 in which case the anode voltages are +90 and +54 and *vice versa* for the two states of the circuit.

The truly bistable flip-flop is shown in *Figure 10.5.*

Here the general circuitry is exactly as in the Schmidt trigger with the addition that control grid G_1 is linked to the anode of V_2 by means of a network identical with that used for connecting the anode of V_1 to the grid of V_2. The effect of this additional path is that if V_1, for example, is conducting, the potential at *b* is low. This is communicated to the grid of V_2 which is thus kept cut off. Since V_2 is cut off its anode potential, at *a*, is high and thus maintains the grid of V_1 at a suitable potential for conduction to occur.

Since the circuit is symmetrical in valves and components, the above argument can equally apply to the case where V_2 is conducting and V_1 is cut off.

The circuit can be triggered from one state to the other by pulses of suitable polarity applied to grids or anodes.

In order to make a binary counting element from the bistable

flip-flop it is necessary to add to it some device which maintains a record of the previous state of the circuit whilst the change-over from one state to the other occurs.

Using the basic circuit of *Figure 10.5* as the bistable element and representing it by the shorthand notation shown, two binary counting elements are given in *Figure 10.6.*

In the first element, *Figure 10.6* (*a*), the 'memory' is provided by the two condensers C_m connecting each anode with the opposite grid. The circuit is triggered by the application of suitable pulses

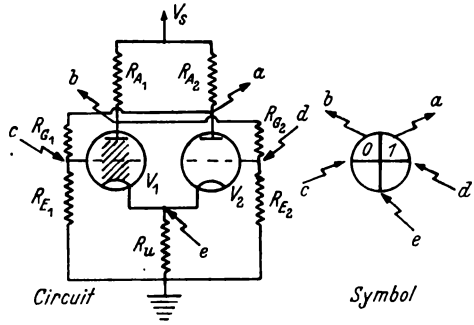


Figure 10.5 Bi-stable flip-flop

to both grids, simultaneously, *via* the coupling elements G_C and input *S*. The elements G_C may take the form of simple condensers, diodes, or triodes, or in fact any elements which do not effect d.c. connection between the two grids.

An alternative point for insertion of the trigger pulse is the common cathode *K*; it is easily seen that any pulse applied here is equivalent to a pair of pulses at the grids.

A simple explanation of the triggering action is as follows. Suppose that negative pulses are applied at the grids; that applied to the non-conducting valve will have no effect, but that applied to the conducting valve will be amplified and fed, *via* the condenser C_m, as a positive pulse to the grid of the non-conducting valve. At this point it will be of sufficient amplitude to swamp the original negative pulse and will cause conduction in the originally non-conducting valve. From this point the action is regenerative since a further negative pulse is sent back to the grid of the originally conducting valve to reinforce the negative pulse already present. The action ceases when the state of the circuit is the reverse of that obtaining initially.

The circuit of *Figure 10.6* (*b*) makes even more evident the memory properties required to convert a bistable flip-flop into a counting element. Here each anode is connected to its own grid *via* resistor R and diode D. Since the anodes are necessarily at higher potential than the grids the diodes normally allow no conduction. The diode cathodes are connected to the pulse line *via* capacitors C. During the non-operative existence of the circuit capacitors C will charge to their respective anode potentials *via* resistors R.

Suppose, now, that a negative pulse is applied at S. This will be transmitted *via* condensers C to the diode cathodes. Suppose side

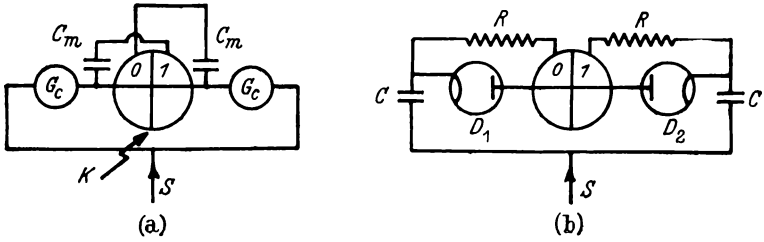


Figure 10.6 Counting elements

0 to have been initially cut off, its anode will be at high potential and this will have been registered at D_1, the negative pulse *via* C is then insufficient to cause conduction of D_1. On the other side, however, since conduction was occurring, the anode potential will have been low and the diode D_2 will have a cathode potential only slightly greater than that at its anode. When the pulse *via* S and C arrives D_2 will conduct and cause a negative pulse to appear at the conducting grid of the flip-flop. This side is thus cut off and regeneration occurs causing the initial state of the element to be reversed.

The storage properties of the *R.C.* networks are clearly seen to be such as to prevent the rapid regenerative action of the flip-flop circuit communicating itself to the input points and thus preventing operation.

Practical details of the above circuits are as follows: $V_1 = V_2 = 6J6$. $D_1 = D_2 = 6AL5$ or 1 N38. $R_{A_1} = R_{A_2} = R_{E_1} = R_{E_2} = 22$ KΩ. $R_{G_1} = R_{G_2} = 33$ KΩ. $R_K = 10$ KΩ. $R = 68$ KΩ. $C_m = C = 50$ μμF. $V_A = 150$ V. A negative pulse of 1 μsec rise time and amplitude between 20 and 60 V is required in *Figure 10.6* (*b*). Operationally this circuit has proved extremely reliable and easy to set up.

A LOGICAL STORE

For machines working in the serial mode and having megacycle repetition frequencies the flip-flop single digit storage devices described above are not suitable. It is true that by the use of power valves with load and bias resistors of magnitude measured in thousands of ohms instead of the tens of thousands given above it is possible to construct flip-flops for use in the megacycle range; this does not, however, seem a satisfactory solution of the problem.

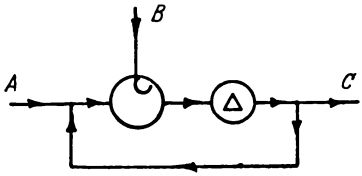


Figure 10.7 Logical single digit store

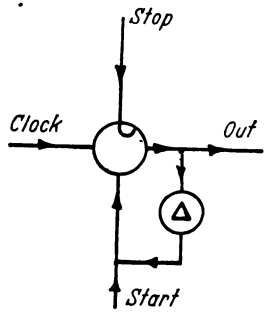


Figure 10.8 More complex logical store

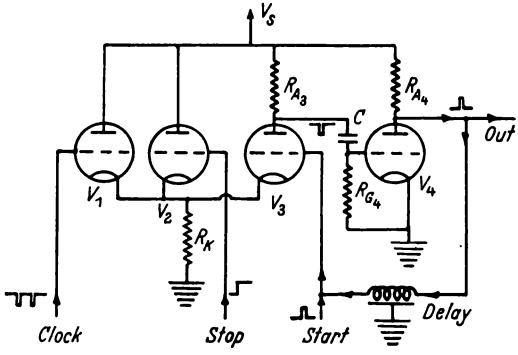


Figure 10.9 Circuit for Figure 10.8

A simpler approach is *via* the purely logical device shown in *Figure 10.7.*

It is seen that in the absence of an input on A the circuit will be quiescent. If now A receives a suitable pulse this will pass through the gate (B being inactive) and emerge after unit pulse delay D at C. The delayed pulse is fed back to A and maintains the cycle

indefinitely. In order to restore the circuit to zero state an inhibitory input is applied at *B*.

This simple device has the disadvantage that it would not keep in step with the main clock pulse source of the machine. A more complicated variant is shown in *Figure 10.8*.

In this circuit the starting and delayed pulses are made to gate a standard set of clock pulses with the result that the store gives an output which is synchronized with the rest of the machine.

A simple way in which the circuit of *Figure 10.8* might be constructed is shown in *Figure 10.9*.

In the circuit V_1, V_2 and V_3 function as a 2 gate with inhibitory input at V_2. The negative going output from R_{A_3} is amplified and inverted by V_4 and is then emitted from R_{A_4} as an output. In addition it passes, after delay by the electromagnetic line, back to the grid of V_3 and thus maintains the cycle.

MAGNETIC (STATIC) STORAGE

The basic ideas of this type of storage will already have been made apparent by the discussion of magnetic gates given in Chapter 9.

Two possible means of magnetic data storage are, first by the use of rectangular hysteresis loop materials, used in circuits of the type shown in *Figures 9.15* and *9.16*, and second, the essentially binary store shown in *Figures 9.17* and *9.18*.

It should be noted that both of these devices can only be read by a process which involves the loss of the contained data. In order to provide continuous storage over an interval embracing many pulses it is necessary to provide some form of regeneration. This can be done electronically, as suggested in the logical gate previously described, a more satisfactory method, however, is to use a shifting register pair of the type to be described in the next chapter.

Single digit storage devices constructed from rectangular loop materials do not, at present, appear suitable for operation at speeds of more than 50 kc/s for the metallic cores and 200 kc/s for ferramics.

MAGNETIC (DYNAMIC) STORAGE

Two forms of dynamic single digit magnetic stores have been suggested. In the first, or magneto-acoustic device, use is made of two properties of magnetic materials:

(1) Remanence. (2) Magnetostrictive effect.

A fine wire of, say, nickel has three small coils of wire *A*, *B* and *C* wrapped on it. *B* and *C* are wound on top of each other and *A* is separated by a variable distance.

If a current is passed through *B* the nickel wire is magnetized with appropriate polarity. If now a current pulse is applied at *A* magnetostrictive contraction (or expansion) occurs in the nickel and a sonic pulse is propagated along the wire. When this sonic pulse

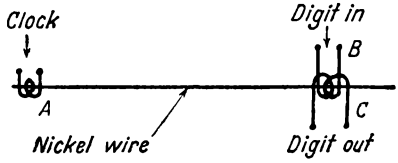


Figure 10.10 Simple magneto-acoustic store

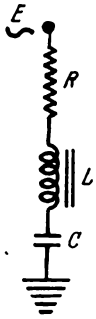


Figure 10.11 Basic ferro-resonant circuit

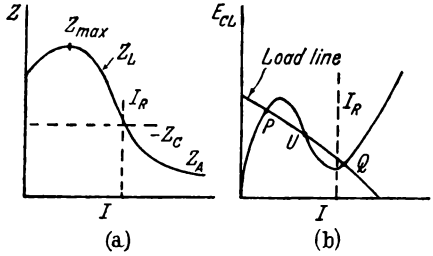


Figure 10.12 Current curves for Figure 10.11

reaches *BC* it causes, in effect, a movement of the elementary magnetized region under the coils and thus, by induction, produces an output voltage at *C*.

This device is robust, simple to construct, capable of megacycle operation and produces adequate output. It has the property that by appropriate spacing of *A* and *BC* the stored data can be read out after delay, instead of immediately as is the case with other devices.

There is no limit to the length of wire which can be used and up to 4 binary digits per inch can be stored. Some further properties and circuit details will be given in Chapter 12.

An alternative dynamic magnetic digit store is the so-called ferro-resonant flip-flop. This makes use of the property of saturable

magnetic materials when operated in the simple circuit of *Figure 10.11.*

When alternating current flows in an iron cored inductor the inductance of the latter first increases with current and then, after reaching a maximum, decreases asymptotically to fixed value. This is shown in the $\mathcal{Z}_L$ curve of *Figure 10.12* (*a*), with a normal magnetic material, of the Mu-metal type. The change between $\mathcal{Z}_{L\ max}$ and $\mathcal{Z}_A$ is of the order of 20:1, and with rectangular loop materials even greater variation is possible.

Consider next the behaviour of the circuit of *Figure 10.11.* It is seen by considering *Figure 10.12* (*a*) that the voltage E_{CL} across the *LC* combination first increases with current, reaches a maximum, and then drops to a minimum at which the capacitor reactance is the negative of that of the inductor, this position corresponding to a

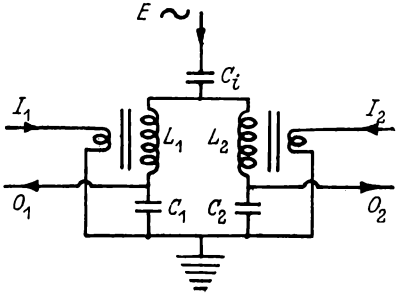


Figure 10.13 Practical form of ferro-resonant flip-flop

resonant condition of the *LC* circuit. Further increase in *I* causes a corresponding increase in E_{CL}.

When a load line is added to *Figure 10.12* (*b*) it is seen that two intersections *P* and *Q* occur at which stable conditions exist. (That at *U* corresponds to a negative resistance characteristic and is unstable.)

The circuit of *Figure 10.11* can thus exist in two stable current states for suitably chosen values of the applied alternating voltage $E\sim$.

A practical form of the ferro-resonant circuit is shown in *Figure 10.13.* Here two resonant units L_1C_1, L_2C_2 are supplied from an a.c. source $E\sim$ *via* a common condenser C_i. C_i and $E\sim$ are so chosen that only one of the circuits can be in resonance at any given time, thus if both are at resonance the voltage E_i will drop until one goes out of resonance, whereas if neither is in resonance E_i will rise until one goes into resonance.

D.c. outputs can be obtained by connecting rectifiers in series with the output leads O_1 and O_2.

The circuit can be caused to take up a given state by means of a pulse applied at either of the windings I_1 and I_2; if both are pulsed simultaneously the circuit will change state and can thus be used as a conventional binary counter.

In a commercially available example of this component a supply frequency of 500 kc/s is used and trigger pulses of 2 μsec duration and 20 V amplitude are adequate for a repetition rate of 20 kc/s.

NON-BINARY STORAGE

The representation of single digits in scales other than binary and at repetition rates of the order of one megacycle per second is a problem which has not yet received an adequate solution. Three devices may be mentioned as having been tested for this purpose:

(1) The Dekatron. (2) The S.T.C.-tron. (3) The Trochotron.

In the first two of these use is made of a gaseous discharge and the speeds are moderate—1 kc/s and 20 kc/s respectively. The third device is capable of operation at megacycle rates but is neither commercially available nor simple enough to be used in the quantities needed in a modern calculating machine.

The Dekatron is a normal sized valve (a miniature version also exists) having a set of four electrodes mounted in an atmosphere of an inert gas of the neon type. The electrodes g_1, g_2, K and K_0 form a cage-like structure surrounding the disc shaped electrode A, shown in *Figure 10.14.*

In operation the anode A is connected to a voltage source *via* a high resistance, the cathodes K are connected directly to earth, K_0 is connected to earth *via* a resistor having a much lower value than that in the anode circuit, and the grid systems g_1 and g_2 are returned to a potential, intermediate between that of the anode and ground. A typical circuit is shown in *Figure 10.15.*

Since the two cathode structures are at considerably lower potential than that of the grids, a glow discharge will start at A and settle on one of the wires of K or on K_0, the high anode resistance making impossible a general discharge at all points. The cathode K_0 is used as a zero point and the discharge can be caused to settle upon it by a momentary removal of the earth connection from K. If now a negative impulse of sufficient duration is applied to g_1, the glow will spread from the wire of K (or K_0) to the adjacent wire

of g_1. A further negative pulse applied to g_2 will share the discharge between g_2 and g_1 and, if the pulse on g_1 is now removed, the discharge will spread to the wire of K (or K_0) adjacent to g_2. Removal of the pulse on g_2 results in the re-stabilization of the discharge on K (or K_0), but advanced one position from its initial state.

It will be seen that this action is precisely that required of the ring counter described in Chapter 6 and also, that a reversal of the order of operations on g_1 and g_2 will cause a backward movement of the glow. The two overlapping pulses which are required to operate the Dekatron are quite simply obtained by the use of an integrating network placed between g_1 and g_2 (or *vice versa*).

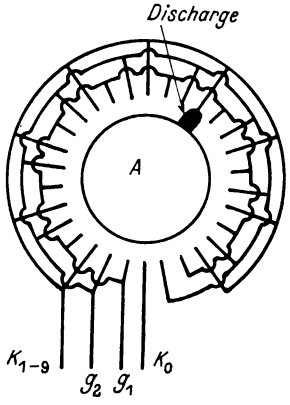


Figure 10.14 Dekatron structure

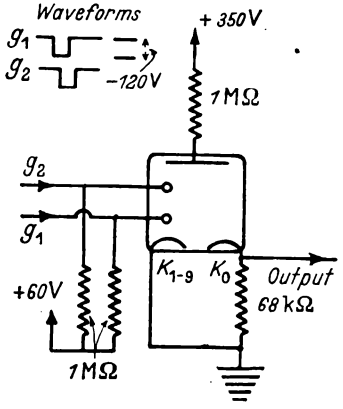


Figure 10.15 Typical Dekatron circuit

A particular advantage of the Dekatron lies in the fact that the position of the glow is visible so that, by the provision of a suitable mask, the digital contents can be viewed at any time. On account of its limited speed of operation (1 kc/s) the Dekatron GC10A has not been much used in automatic computers; a faster variety of cold cathode discharge tube for decimal ring counting has been evolved by the *Standard Telephone and Cables Laboratories* and this, with its 20 kc/s operating speed, may well find a place in the medium speed computing installations of the future, its type number is G10/240E.

The Trochotron is an electron beam tube which is such that the beam can take up any one of a set of stable positions and can be stepped by the application of suitable pulses. Since it does not depend on gaseous discharge it is capable of great speed of operation, it is very complex in structure, however, and will not be discussed further here.

The devices described above are normally constructed for operation in decimal scale but there is no reason why they should not be produced with any given number of electrodes. Versions of both the GC10A and G10/240E are under development in which all nine cathodes are brought out; this would have the advantage of allowing the contents to be sensed statically, without cycling the tube and observing the passage of the discharge through the zero position by the elevation of the potential of K_0 (*Figures 10.14* and *10.15*). Auxiliary pulse and carry storage tubes are available for the S.T.C.-tron or G10/240E, these also have cold cathodes and are economical in power requirements, their type number is G1/370K.

11

MISCELLANEOUS COMPONENTS

IT IS the purpose of this chapter to deal briefly with a number of computer components which, although their form and detail is readily predictable from the previous discussion, are of major importance in the overall design of a computer. Most of these devices will by now be familiar to the reader, in essence at least, from their mention under the relevant logical discussions. Typical of such units are the shifting registers whose properties have been defined in the discussion of adding and multiplying circuitry. In this chapter such units will not be discussed if they are identical in form and construction with the main machine storage; this is particularly the case for acoustic registers, or 'short tanks' whose construction is identical with that of the main acoustic store discussed in Chapter 12. It also applies to registers using the storage properties of secondary emissive media of the cathode ray tube type and of recyclic magnetic recording techniques.

Other components are less dependent on the structure of the particular machine in which they are used, and of this class typical examples are function tables and coincidence units.

SHIFTING REGISTER UNITS

It is proposed to discuss two general types of register unit, that based on valve single digit storage elements and that based on the rectangular hysteresis loop magnetic elements.

In the class of valve registers the fundamental problem is that of transferring the contents of a register stage A_m (*Figure 11.1*) into stage A_{m+1} of the same register and performing identical operations on all stages of the register. Three general methods of achieving this suggest themselves. In the first A_m is connected to A_{m+1} by means of a gate G_m which has the property of retaining the state of A_m for a short time after its contents have, in fact, been altered, and of passing the state of A_m on to A_{m+1}.

The second method is to clear all stages to zero and to arrange the circuits in such a manner that if any stage contains unity it

emits a pulse on restoration to zero. This pulse is delayed by $\triangle_m$ and then resets A_{m+1}, say, after the latter has had time to recover from the previous clearing operation.

The third method is a strictly logical one. The contents of all stages of A are transferred to an auxiliary register B, initially clear. A is then cleared and the contents of B sent back to it displaced by one digital position in the desired direction.

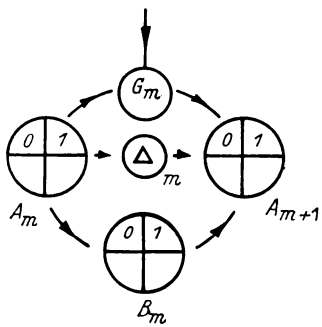


Figure 11.1 Types of register

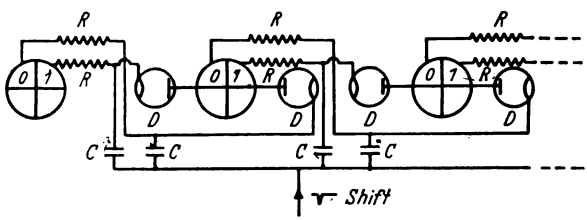


Figure 11.2 Typical delay coupling register

Registers of each of the above types have been built and operated and, except on grounds of complexity, there is little to choose between them.

A typical register of the first type is shown in *Figure 11.2*. In this, the anodes of the projecting register stages are connected, *via* resistors R and diodes D, to the grids of the receiving stages. Current stage contents are effectively stored on capacitors C with time constant RC. D.C. isolation between anode and grid circuits is normally maintained by diodes D. Application of a suitable negative shift pulse to the common condenser line results in the application of a negative pulse, *via* the relevant diode, to that grid of the receiving register stage connected to the conducting (low potential) anode of the projector. The effect of this is to set the

receiving stage to the same digit as was originally contained by the projector.

All of the stages change simultaneously and memory of the initial contents during the application of the short shift pulse is *via* the time constant *RC*. Suitable values are: $R = 68{,}000\ \Omega$. $C = 50\ \mu\mu F$.

Circuitry for the $\triangle_m$ type register is illustrated in *Figure 11.3.* Here all register stages are cleared to zero state by means of a negative pulse applied to their grids. Those containing unity will emit, at their left-hand anodes, a negative step voltage; this is delayed by the electromagnetic line and, when the clear pulse and subsequent disturbances have died away, passes *via* isolating condenser *C* and diode *D* to reset the next flip-flop to unit contents.

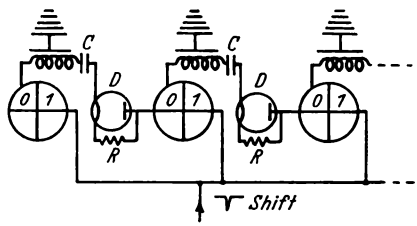


Figure 11.3 Typical delay line 'clear' register

Resistor *R* is needed to bias the diode and can be omitted when germanium elements are used.

Registers of this type appear to possess no advantages whatever over the directly gated variety previously described. They have also the considerable disadvantage of requiring expensive electromagnetic delay elements which are not commercially available.

A good example of a strictly logical shifting register is shown in *Figure 11.4.*

In this circuit a triode section, $V_{n-1,1}$ has its grid potential set by that of flip-flop A_{n-1}. The cathodes of all such triodes (one for each stage A_m) are connected to a common conductor K_1 whose potential is normally maintained at a voltage above that of the A_m grids. Under these conditions triode $V_{n-1,1}$ is cut off. Suppose, however, that the cathode potential at K_1 is progressively lowered; at some point $V_{n-1,1}$ will conduct and thus set B_{n-1} to the state in which its right-hand anode is conducting. If the potential of K_1 is still further reduced, $V_{n-1,1}$ will eventually draw grid current and thus forcibly reduce the grid potential of A_{n-1} causing the latter to be set into the zero state (i.e. with its right-hand element cut off).

The cycle of operations described above would result in the invariable clearing of all A stages and the setting to zero of all B stages. If, however, K_1 has its potential reduced less drastically, $V_{n-1,1}$ will conduct only if the right-hand grid of A_n is initially at high potential, and not at all if it is already at low potential. The effect of this is

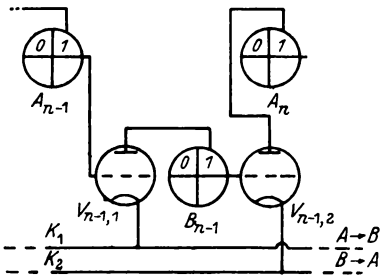


Figure 11.4 'Logical' register

to transfer any 1 stored in A_{n-1} into B_{n-1} and to clear A_{n-1} but to leave B_{n-1} zero if A_{n-1} is itself zero.

By duplicating the above circuitry the contents of B_{n-1} can be similarly transferred into A_n. The effect of this combined operation is effectively to shift the contents of A one place to the right.

A positive advantage of this circuit is the complete absence of capacitors in its construction. The consequence of this is that

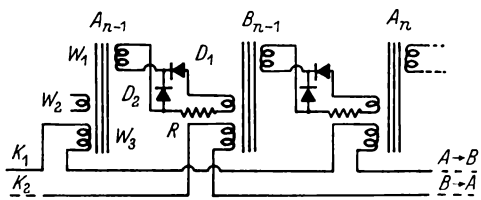


Figure 11.5 Magnetic shifting register

circuits of this type can be operated at very high speeds, for example, the 2 Mc/s shift rate attained in the Princeton computer. The addition of the elements B is made use of in this machine, to facilitate the processes of multiplication and division.

A disadvantage of the circuit lies in the low impedance of the sources needed to drive K_1 and K_2. In practice stages must either be driven in distinct groups from several high power cathode followers or alternatively several parallel valves have to be used as a single cathode follower.

A shifting register on the same principle as that just described may be constructed from rectangular loop magnetic elements. One variant is shown in *Figure 11.5.*

Energy is stored in core A_{n-1} by saturating it magnetically *via* winding W_2, say, and allowing it to relax to a state governed by its remanence. A current pulse is now applied to W_3 in such a sense that the magnetic state of A_{n-1} is reversed. Under these conditions an induced voltage is produced across W_1 and this passes *via* rectifier D_1 and resistor R to a receptor winding on the intermediate core B_{n-1}. Supposing that the remanence of the cores A, B is sufficiently high, it can be shown that the core B_{n-1} can be driven to saturation despite the inevitable circuit losses. A crude demonstration of this is the following: suppose that a flux change $\mathrm{d}\varphi_a$ is produced in core A and that winding W_1 has $\mathcal{N}_1$ turns whilst the receptor winding W_2 on B has $\mathcal{N}_2$ turns. Then, assuming a flux change $\mathrm{d}\varphi_b$ to occur in B, the differential equations of voltage are:

$$\text{Output from } W_1 = \mathcal{N}_1 \frac{\mathrm{d}\varphi_a}{\mathrm{d}t}$$

$$\text{Input to } \quad W_2 = \mathcal{N}_2 \frac{\mathrm{d}\varphi_b}{\mathrm{d}t}$$

or, since these are necessarily equal:

$$\mathcal{N}_1 \frac{\mathrm{d}\varphi_a}{\mathrm{d}t} = \mathcal{N}_2 \frac{\mathrm{d}\varphi_b}{\mathrm{d}t}$$

Integrating:

$$\varphi_b = \frac{\mathcal{N}_1}{\mathcal{N}_2} \varphi_a + \text{const.}$$

Thus, neglecting circuit losses, φ_b can be made greater than φ_a by any desired amount simply by varying the ratio $\mathcal{N}_1/\mathcal{N}_2$. In practice this is not true and a more elaborate analysis, taking into account circuit losses, shows that the optimum energy transfer value is obtained when $\mathcal{N}_1 = 2\mathcal{N}_2$.

The purpose of the diodes D_1, D_2, and the resistor R is to prevent the back flow of current from A_{n-1} to B_{n-1} *via* W_2.

For materials having adequately rectangular hysteresis loops, the output from A_{n-1}, when its initial magnetic state is identical with that generated by the current pulse in W_3, is insufficient to produce any change in the state of the following stage B_{n-1}. Thus any digital pattern stored in the A cores will be transferred to the B cores on application of a pulse to the windings W_3 *via* K_1. The A

cores are then left in the cleared or zero state and thus receive the pattern from B upon application of a similar transfer pulse to the W_3 windings of the B elements *via* K_2.

Practical details of the above circuit are as follows: the rectangular loop material should consist of 2 or more turns of ·001 in. × $\frac{1}{8}$ in. strip wound in a 'clockspring.' The diameter of the ring thus produced should be ·375 in.–·50 in.

It is desirable that the remanence of the material should exceed 90 per cent of its saturation flux, as a practical criterion. It has been shown by the *Harvard Computation Laboratory*, however, that a more correct evaluation of suitability is

$$\frac{\mu_o}{\mu_s} > \frac{8H_s}{H_r}$$

where the various quantities are defined in *Figure 9.16* (p. 87).

The winding scheme should be: $W_1 = 150\ t$, $W_2 = 75\ t$, $W_3 = 200\ t$, and under these conditions a set of 32 coils can be series connected

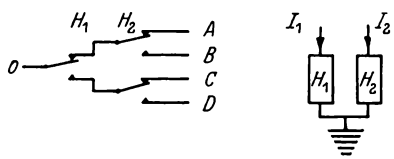


Figure 11.6 Relay tree function table

and driven from any pentode type valve capable of delivering a 20 μsec 125 mA current pulse (e.g. the 6V6).

Resistor R should be approximately 15 Ω and the rectifier units should have not more than 6 Ω forward resistance and not less than 1,000 Ω back resistance at an applied d.c. potential of 20 V.

Units of this type constructed from *Deltamax*, *H.C.R.* or *Permalloy F* alloys have given excellent reliability at shift frequencies ($A_{n-1} \rightarrow A_n$) of 30 kc/s.

FUNCTION TABLES

The two types of function table, one-many and many-one, have already been defined in Chapter 5. It will be remembered that the many-one type is required to produce single unique outputs in accord with any particular combination of several inputs which may be selected.

Possibly the simplest and most reliable device of this kind is the relay 'tree' shown in *Figure 11.6*. The circuit is drawn to illustrate

the case of two inputs I_1 and I_2 which may be energized in any combination. I_1 is connected to the coil of a relay H_1 having a single change-over contact, whilst I_2 is similarly connected to the coil of a relay H_2 having a pair of change-over contacts. The contacts of H_1 and H_2 are connected as shown and it will be seen that the line O is connected to the four outputs A, B, C and D in accord with the scheme:

Table 11.1

H_1	H_2	O
0	0	A
0	1	B
1	0	C
1	1	D

The scheme can, of course, be extended to larger numbers of inputs and corresponding increase in output selection. In general, for n inputs the relay H_n must have 2^{n-1} change-over contacts.

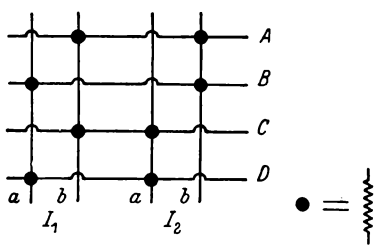


Figure 11.7 Matrix function table

The relay tree circuits have the advantage of producing 100 per cent discrimination between the selected output and the $(n-1)$ rejected ones; they have the disadvantages of limited speed, (1 msec maximum) and comparatively high cost for large numbers of inputs.

The type of many-one function table which has found most frequent use in electronic computer developments to date is shown in *Figure 11.7.*

Here the inputs I_1, I_2 are assumed to be in the form of pairs of voltages a and b such that, for example:

$$I_1 = 0 \quad \text{has} \quad a = 0\ V \qquad b = +100\ V$$
$$I_1 = 1 \quad \text{has} \quad a = +100\ V \quad b = 0\ V$$

and similarly for I_2.

The output lines A, B, C, D are connected, as shown, to the $I_1(a, b)$ $I_2(a, b)$ wires by means of resistors R. It is readily seen that the voltages at $A \ldots D$ for differing combinations of I_1 and I_2 are:

Table 11.2

Line	*Voltage*			
	$I_1=I_2=0$	$I_1=0, I_2=1$	$I_1=1, I_2=0$	$I_1=1, I_2=1$
A	+100	+ 50	+ 50	0
B	+ 50	0	+100	+ 50
C	+ 50	+100	0	+ 50
D	0	+ 50	+ 50	+100

Thus for each combination of inputs there exists only one output line whose potential is 100 V (or for that matter 0 V). The device can therefore be made to operate a gate at the selected output only.

In practical applications it is necessary to use relatively high values for the resistors R, greater than ·25 Megohm for example. This limits the speed of the output pulse rise to a value governed by the time constant of R and the external capacitance. More serious, however, is the fact that when the number of inputs n is increased, the discrimination between the voltage of the selected output and that of its nearest neighbours decreases as $1/n \times$ voltage change across (a, b). This means that very high (a, b) differentials have to be used if more than, say, 5 inputs are required.

The latter disadvantage can be overcome by replacing all resistors except those in column I_1 by diode elements whose forward resistance is much less than R_1 and whose back resistance is much greater. With this modification the output voltages become sensibly b for the selected channel and a for all others.

The only real disadvantage of the diode matrix is its expense, and this may, with a certain loss of speed, be reduced by the use of selenium disc rectifiers.

For outputs which do not require direct currents, alternative schemes are available which make use of saturated core inductors; one such device is shown in *Figure 11.8*. In this a number of rings of rectangular loop magnetic material are wound with 4 coils. The a and b lines of I_1 and I_2 are now arranged either to carry current (state b) or not (state a). For the condition shown all cores, except A, have at least one coil passing current and are, in consequence, saturated. To detect the operative coil a pulse current is applied to all rings *via* the series winding p. This

current is so arranged that only when *no* current flows in another winding of a particular ring does the magnetic state of that ring reach the point of change (cf. *s* in *Figure 9.16*). In this one ring, however, a large flux change occurs and by normal transformer action an output appears at *A*.

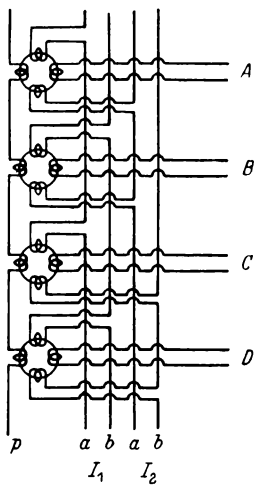


Figure 11.8 Saturated reactor function table

A considerable advantage of this type of function table lies in the fact that large currents can be obtained at the outputs by the simple means of making *A*, *B*, *etc.* step down windings with respect to *p*.

COINCIDENCE SENSERS

In computing machine design it is frequently necessary to have available means for ascertaining the identity of two quantities. A pair of simple relay circuits for performing this operation are shown in *Figure 11.9*. The first design (*i*) is a straightforward function table in which those outputs corresponding to identity between the two inputs are joined. It is clear that certain contacts are redundant in this circuit and a moment's reflection leads to the variant (*ii*). If a relay be available having two identical coils a still simpler circuit is shown in (*iii*); when neither coil is energized the relay stays in the off position as shown, when either coil is energized separately the relay operates and the circuit is broken, when both coils are energized together they are arranged so that the resultant fields cancel out and the output contact remains closed.

When a fast coincidence circuit is required analogy with *Figure 11.9* (*i*) suggests that a simple matrix function table would perform the operation. This is obviously true and a comparison with *Figure 11.9* (*ii*) again suggests the variant shown in *Figure 11.10.*

In this circuit the potentials at the cathodes of D_1 and D_2 are only both median when either $I_1 = 1$, $I_2 = 1$ or $I_1 = 0$, $I_2 = 0$. Thus, only in this state can the potential at p rise above the lowest diode

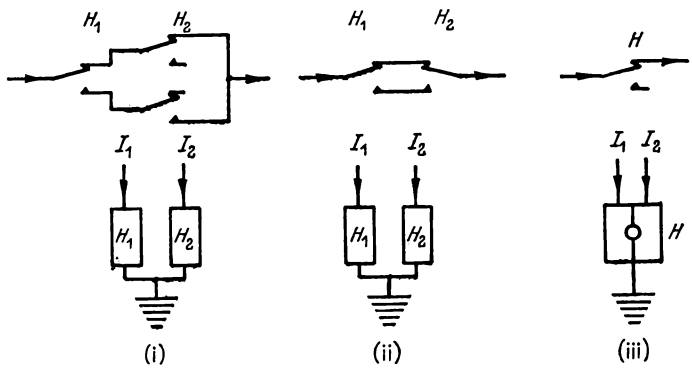


Figure 11.9 Relay coincidence sensers

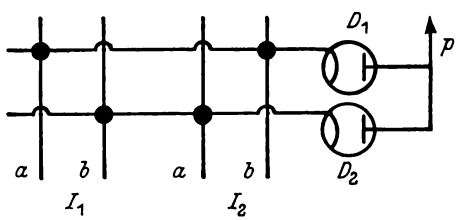


Figure 11.10 Coincidence sensers

cathode voltage. The characteristics of this circuit may be improved by replacing resistors R_2 by diodes in which case the coincidence output at p is equal to the voltage difference between a and b instead of one-half as in the strictly resistive case.

An obvious extension, when the identity of two digit patterns is to be established, is to compare the individual pairs of digits by means of circuits of the above type and to connect the outputs to an n gate of the type discussed in Chapter 9. In relay circuits, however, the even simpler technique of series connection of circuits of the type shown in *Figure 11.9* (*ii*) and (*iii*) is perfectly adequate.

12

STORAGE DEVICES

POSSIBLY the most important component of a modern computing machine is that which performs the operation of storing digital data. This has frequently been called the 'memory' and to some extent this title is justified since storage devices have many features in common with the memory processes of animals.

The functions required of a computing machine store are four in number.

(i) It must be capable of storing incident digit patterns for an indefinitely long time.

(ii) It must be capable of emitting such data when called upon to do so.

(iii) It should be inexpensive.

(iv) It must be possible to erase stored data and to substitute new.

In addition, the speed at which data can be inserted or recalled —usually called the 'access time'—should be comparable with the unit operation time of the arithmetic and control circuits with which the store is to be used.

At the present time it may be asserted that there is no device which satisfies all of the above desiderata perfectly and it is proposed, in the present chapter, to discuss the various solutions which have been constructed and to comment briefly upon their virtues and defects.

A catalogue of suggested or existing storage devices reads like a science degree syllabus in physics, for heat, light, sound, properties of matter, mechanics, electricity and magnetism have all found their advocates and almost all have been brought to a state of physical existence. It is proposed as far as possible to deal with these devices in order of speed rather than in any sequence dictated by physical properties. Even within this framework it is impossible to be consistent since certain techniques, by means of alteration to the logical structure of the machine, can be made to compete in speed of access with fundamentally more rapid devices.

ELECTROMECHANICAL STORAGE

Direct storage of data on relays is possible but not practicable where a large capacity is required. Several devices have been suggested, however, which do have high volumetric and financial efficiency and it is proposed to describe some representative specimens from this group.

The first type consists of a circular plate which can be rotated about an axle *a* through its centre and normal to its plane.

Around the periphery of the disc are a number N of equispaced holes *h* in each of which is a small pin. The pins can be pushed,

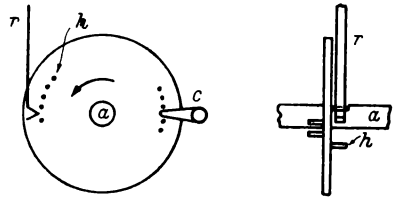


Figure 12.1 Pin store

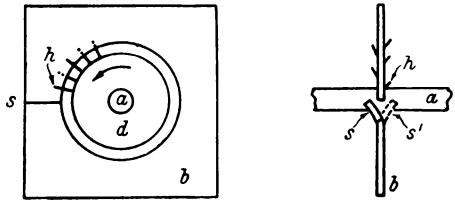


Figure 12.2 Wire store

by means of a solenoid operated mechanism *c*, to either side of the disc, and the latter can be rotated so as to stop with any particular pin either under the pusher *c* or under a read out contact *r*. Only those pins projecting from the same side of the disc as *r* will make contact and consequently the device can function as a binary storage unit.

Both serial and parallel operation is possible, in the first, the pins of a single disc are brought, in sequence, past the reading contact, whilst in the latter corresponding pins in a number of discs rigidly mounted on a single shaft are sensed.

The volumetric efficiency of the device is quite high considering its mechanical nature, a typical example stores 50, 21 binary digit numbers in a cylindrical volume 8 in. long by 2 in. dia.

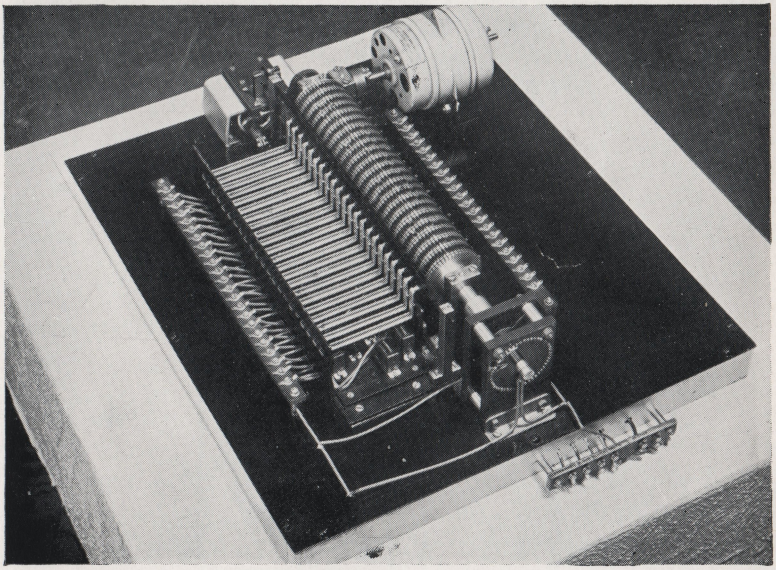


Plate 2. Electromechanical Store

In practice more sophisticated means have to be employed to prevent vibrational displacement of the pins during operation; these, however, are of an obvious nature and need not be discussed here. A photograph of an actual store of the above type is shown in *Plate 2.*

A similar device, but capable of greater ease of manufacture, is shown in *Figure 12.2.* Here the storage elements take the form of thin wires *h* projecting radially from a metal disc. These wires can pass on either side of a thin partition *b via* a slot *s* and a flexible tongue of metal. When the tongue is in the plane of the partition the wires are unable to change from one side of the partition to the other and thus store any information permanently. When any digit wire is required to change its contents the inner disc *d* is rotated, and just before the appropriate wire reaches the gating position the tongue *s* is deflected to guide the wire to the required side. Equilibrium is restored before the arrival of the next wire and in consequence all other data is left unchanged. Read out is effected by causing those wires projecting on one side of *b* to pass over an electrical contact.

This device, on account of the simple mechanism used to deflect the wires, is capable of considerably greater speeds than the pin store previously described; it has also the advantage of allowing the disc and wire assembly to be produced by a simple diecasting operation.

In parallel operation both of the above stores have mean access times of the order of ·25 sec, for serial operation the wire store can operate at a frequency of about 50 c/s.

DRUM TYPE MAGNETIC STORAGE

The satisfactory development of magnetic sound recording equipment inevitably led to the consideration of this medium for digit storage. The use of coated tape or wire, whilst eminently suitable for input/output equipment in computer design is, however, hardly suitable for an internal high speed store.

To overcome the handling difficulties associated with magnetic tape and wire it has proved satisfactory to apply the magnetic medium to the surface of a cylinder so arranged as to rotate with great accuracy between a pair of bearings.

Mounted close to the drum are a series of read/record heads consisting of a closed magnetic circuit having a small gap at one point. When a direct current is passed through a coil of wire

encircling the magnetic core a field is produced which spreads to the air in the region of the gap.

Assuming that the drum is stationary, this leakage field will magnetize an elementary portion of the drum surface in the region of the gap and with a polarity governed by the direction of current flow in the input coil.

Suppose, now, that the drum is rotated at high speed, as the elementary magnetized region passes the gap flux will change in the head circuit and an electric potential will be induced in the

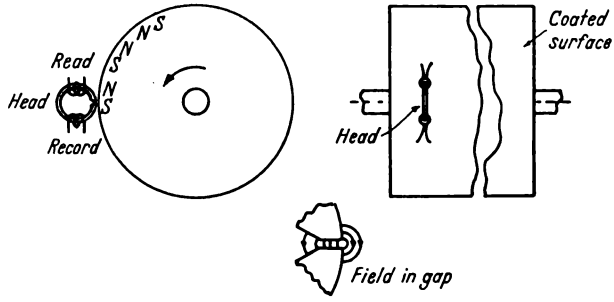


Figure 12.3 Drum type magnetic store

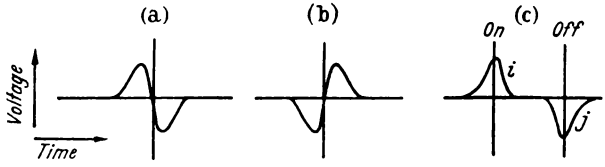


Figure 12.4 Output waveforms from drum

windings. The form of this potential is as shown in *Figure 12.4* (*a*) and (*b*).

According to the direction of the initial magnetization the potential wave will take one of the two forms shown, and it is an easy matter to design circuitry to distinguish between them and set up a flip-flop into zero or unity state. Thus far it has been assumed that the drum was stationary whilst the original act of magnetization was being performed, if this were not the case a slightly modified waveform results; this is shown in *Figure 12.4* (*c*). The presence of the flat between the two terminal peaks *i*, *j* is due to the fact that over the region *i*–*j* the flux density of magnetization is constant and in consequence no voltage is induced in the reading head.

In order to represent a series of binary digits by means of the flux pattern on a drum of the above type, a number of different methods of magnetization are possible; these correspond in general type to the well-known techniques of signal transmission used in telegraphy. Chief among the methods are the following:

(a) Short pulse recording.
(b) Rectangular pulse recording.
(c) Non-return to zero recording.
(d) Amplitude modulated carrier.
(e) Frequency modulated carrier.
(f) Phase modulated carrier.

The waveforms for input and output voltages in each of the above methods are shown in *Figure 12.5.*

In the first section a_i shows an input pulse which is so short that the drum surface may be considered stationary during the recording process. Assuming that the head gap is small the output waveform shown in a_o will be produced. The flat at x is present only if the drum speed is sufficient for the travel between pulses to be greater than ·01 in.

At b_i is shown a somewhat longer input pulse system and at b_o the resulting output.

c_i shows a so-called non-return to zero waveform for the same digit pattern. The output c_o is characterized by having a voltage change only where a change in digit occurs. This system has approximately double the digit packing potentialities of (a) and (b) and has considerable advantages when the design of output systems is contemplated. It has the grave defect, however, that the recording head driving circuits must be capable of maintaining a d.c. saturating current through the head continuously, and this in turn implies the use of high impedence heads. To a lesser degree this defect is shared by the system shown in (b) and is only removed when the short pulses of (a) are used.

Each of the above systems is self erasing, that is, input data automatically erases data already present. The carrier type systems shown in (d), (e) and (f) do not have this property. In d_i is shown an amplitude modulated carrier and in d_o the resulting output; similarly in e_i and e_o are given the waveforms for frequency modulation. On the scale shown both of the above systems are feasible; however, it is evident that the systems are inefficient since several oscillations of the carrier are needed to establish the recorded digit, whereas on the same scale each oscillation space could be used as a recording position if the waveform of (a) were used.

A more satisfactory scheme is that shown in f_o. Here phase modulation of the carrier is used and an efficient system results; some of the virtues of the non-return to zero method are present with the added attraction that, since a carrier is used, the voltage

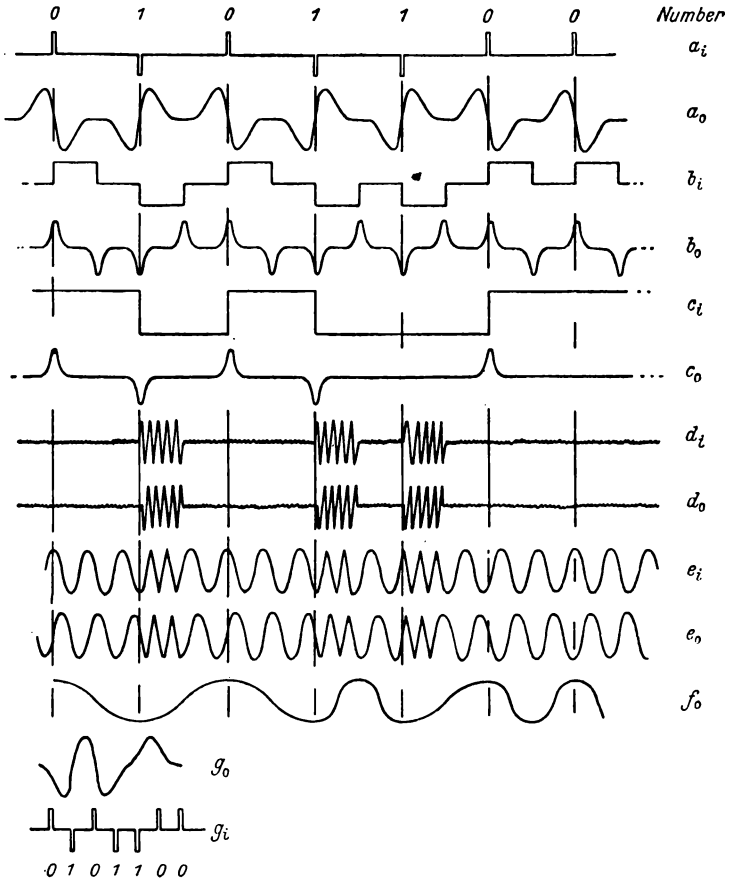


Figure 12.5

waveforms can be passed through a transformer, thus avoiding the necessity for high impedence heads.

Possibly the most attractive scheme of all is shown in g_o. Here the input is again in pulse form; the recording gap is long, however, compared with the physical digit spacing on the medium. The

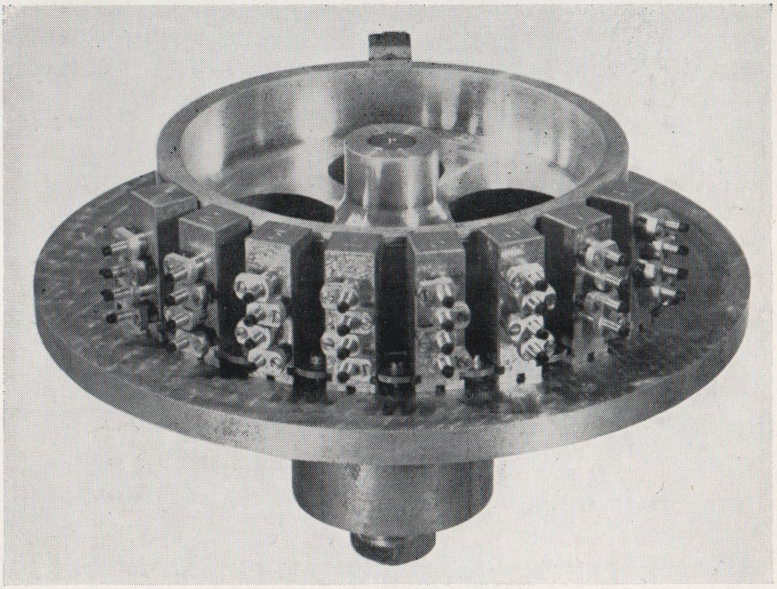


Plate 3 (a). Magnetic Drum Store

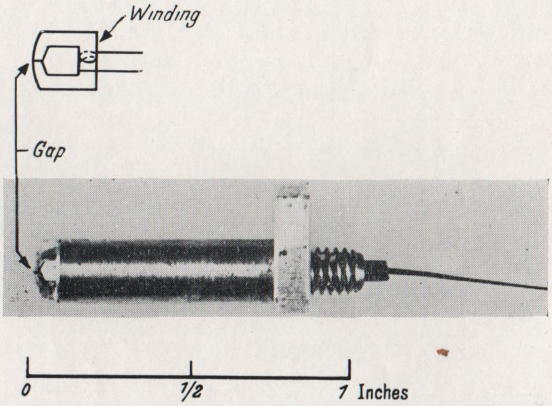


Plate 3 (b). Read/Record Head

effect of this is to compress the waveforms of a_i and a_o so that the flats x vanish and the head of one pulse overlaps the tail of the previous one. The result of this is shown in g_o and it is seen that an essentially non-return to zero output is produced. Naturally the recording gap is not lengthened, but the digit packing density is increased until the effect shown is produced with a standard gap. This uses the medium in the most efficient manner, and in experiments recording densities of more than 200 binary digits per inch have been obtained.

When using magnetic drum storage it is usual to have a number of recording tracks side by side, the system may then be used, either as a parallel access store having separate recording and output arrangements for each track, or as a serial store having one recording pulser and output amplifier which is switched from channel to channel as required.

A typical magnetic drum storage unit, which is available commercially, is shown in *Plate 3 (a)*. The storage capacity of this equipment is 1,024 words, each of 32 binary digits, and it will be seen that, since the surface area is only 15 in.^{2}, a high volumetric efficiency is attained. The type of head used for the read/record operation is illustrated in *Plate 3 (b)*; it consists of a single ·015 in. lamination of high-permeability steel, provided with a working gap of about ·0015 in. Windings may vary from 5 turns, for a low impedence head, to 50 turns, for a head with which the use of a transformer is not desired.

Modern practice would favour the use of a ferrite to replace the steel lamination in head design. Such materials have low eddy current losses and a permeability which can be as great as 1,000 at frequencies in the megacycle range. Unfortunately, thin sections of the ferrites are very brittle, and it has not, to date, proved possible to produce the large numbers of units required. The use of a thick, non-magnetic, ceramic backing would obviate this trouble, but such composite materials have yet to be produced.

For input/output applications, where a head width of ·050 in. is acceptable, ferramic heads have been successfully applied. They have the virtue that, because of their hardness, they show no appreciable wear after prolonged rubbing contact with the tape or wire.

The high currents (5–20 A) required for low impedence heads make the use of a transformer essential and a reasonably economical system, in which the transformer serves both for input and output, is shown in *Figure 12.6.*

The essential feature lies in the use of a centre tapped transformer both as an input device and to provide a balanced output to the main amplifier.

Several methods of track selection are possible when serial working is required. The simplest is the interposition of a relay tree at H, this will give selection times of the order of 2 msec. If faster operation is required electronic gating can be applied either to the

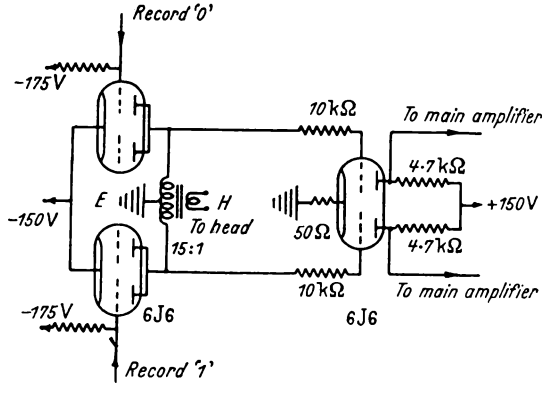


Figure 12.6

amplifier elements directly or, more efficiently, by the introduction of a very high impedence at E.

A suggested logical design for a serial store using the magnetic drum is shown in *Figure 12.7*.

It is assumed that two tracks of the drum D are devoted to pre-recorded 'clock' markers. T_d contains groups of n markers (n = number of digits in word) separated by gaps. T_w has a single marker coincident with each of the gaps in T_d. Data to be stored on the drum is located opposite the markers of T_d. It will be assumed that the particular digit track T_p is selected by means of a relay switch s.

The digit markers from T_d are amplified by A_2 and applied to gate G_2 which is normally held closed by flip-flop F_2. Word markers from T_w, are amplified by A_1 and fed to counter C. The contents of C are compared with the required track position—held in register ML—by coincidence senser CS. Some time during each drum rotation CS will emit a pulse to indicate the start of the required word.

To use the store, flip-flop F_1 is set up, this opens G_1 so that the next coincidence pulse emitted by CS is transmitted and sets up F_2.

F_2 is normally cleared by each word start pulse from A_1 but the circuit delays are such that it is ready to be re-set by the pulse emitted from *CS via* G_1.

This somewhat elaborate system is necessary to ensure that the circuits shall not be switched on except at the start of a word.

The excitation of F_2 opens G_2 to pass the digit markers emitted from A_2, these are then applied to gates G_5, G_6, G_7 and G_8.

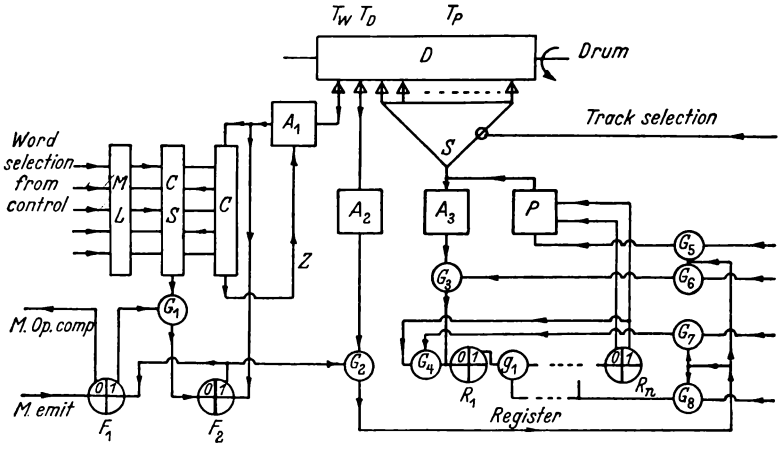


Figure 12.7 Schematic of serial magnetic store

Suppose first that data is to be recorded. The digit pattern is stored in register stages $R_1 \ldots R_n$ and gates G_5, G_7 and G_8 will have been primed by the record instruction circuitry. G_7 and G_8 will emit the clock pulses from G_2 to G_4 and the register shift circuitry thus causing the contents of R to shift at each clock position. At the same time G_5 will emit clock pulses which cause the contents of R_n to be recorded on the drum *via* pulser P.

At the end of the recording operation the register contents will be unchanged and will have been inscribed on the drum, and F_2 will be restored to zero state by the next word start pulse emitted by A_1. The re-setting of F_2 closes G_2 and resets F_1 which in turn emits an operation complete pulse to the rest of the machine.

The read operation is performed in a similar manner except that G_5 and G_7 are not operated whilst G_6 and G_8 are. This means that existing register contents are shifted out and stored data is inserted *via* A_3 and G_3 into R_1 and, by virtue of the shifting properties of R_1 into the remaining stages of R. The operation is terminated in exactly the same manner as that described for recording.

It will be noted that no provision has been made for ensuring that the counter C is correctly set in relation to the markers on the drum. This is conveniently achieved by means of the connection Z. At the most significant track position two pulses are recorded close together instead of the normal one. If the counter has the wrong relationship to the track positions this extra pulse will advance the counter content one step per revolution until parity is attained. At this point the digitally most significant counter stage emits a pulse *via* Z, which paralyses A_1 for a short time thus deleting the extra pulse and resulting in stable conditions.

OTHER TYPES OF DYNAMIC MAGNETIC STORAGE

It is not necessary to disperse the magnetizable medium over the surface of a drum; other possibilities include endless tapes of coated paper or plastic, or of special steel as in the original *Blattnerphone*. A possible means of obtaining a larger output than occurs

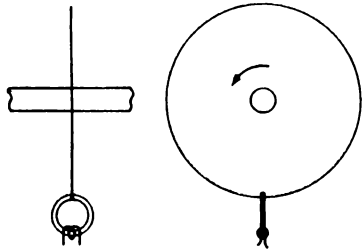


Figure 12.8 Disc type magnetic store

with the drum device previously described is by using a disc of remanent magnetic material rotating between the actual pole pieces of the recording head as shown in *Figure 12.8*. The advantage of this scheme lies in the fact that all of the flux generated during recording passes through the medium instead of the leakage flux normally used. On the other hand, calculation shows that the improvement to be expected from the use of this method would not be much greater than 2:1 and this appears hardly worth while in view of the mechanical difficulties of the arrangement.

It is, perhaps, worth mentioning that a disc type storage system was originally used on the *A.R.C.* calculator, oxide coated paper formed the medium and centrifugal force was used to keep the discs flat and in contact with the recording heads.

Several alternative schemes exist for ensuring optimum use of the recording flux but since these generally lead to loss of volumetric efficiency they will not be described here.

MAGNETIC–ACOUSTIC STORAGE

An extremely simple type of storage device having a multiplicity of uses, both serial and parallel, is shown in *Figure 12.9.*

A rod, tape or wire R, of any material having high remanence and exhibiting magnetostriction, has wound upon it a number of coils A, D_1, D_2 . . . D_n each separated by an appropriate gap. If currents are passed through D_1 . . . D_n in one or other of the two possible directions, that portion of R affected by the resulting field will become magnetized with either NS or SN polarity. On removal of the polarizing currents through D_1 . . . D_n remanent magnetism of the same polarity will remain in the portions of R underlying D_1 . . . D_n.

If now a current pulse is applied to A, because of the magneto-strictive property of R, a compression (or expansion) wave will be

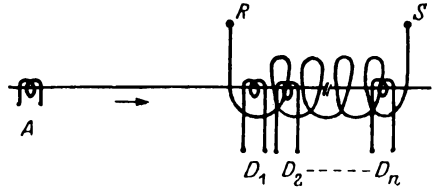


Figure 12.9 Magneto-acoustic storage

generated at A and will travel along R. As this pulse passes under D_1 . . . D_n the elementary movement of the magnetic fields so produced will induce voltages in D_1 . . . D_n whose variations, for a short pulse at A, are shown in *Figure 12.10.* It will be observed that the waveforms are, as expected, identical with those produced by the drum type magnetic store.

Several different input/output arrangements are available. In the scheme just described both input and output are parallel *via* D_1 . . . D_n, alternatively a long coil S may overlay D_1 . . . D_n in which case a serial output will occur *via* S as the sonic pulse passes under the magnetized regions D_1 . . . D_n.

Any digit repetition rate (up to a fixed limit) may be obtained by varying the spacing between D_1 . . . D_n and any desired read out delay by adjusting the distance A . . . D_1.

From the practical point of view R can consist of a ·005 in. dia. nickel wire, the coils $A_1 D_1 \ldots D_n$ having each 10–500 turns, and the minimum distance $D_1 \ldots D_2$ being about ·25 in.

The pulse current through A should be about 2 A and the pulse duration ·1–·5 μsec. To bring the output from $D_1 \ldots D_n$ to a level of 20 V three stages of pentode amplification are required for 10 turn pick-up coils.

In order to eliminate end reflections from the wire it has been found adequate to clamp the ends of the latter between soft rubber sheets, or more simply, to embed the ends in lumps of 'plasticene.'

Since the attenuation of sonic waves in metals is small, very considerable lengths of wire may be used without trouble due to

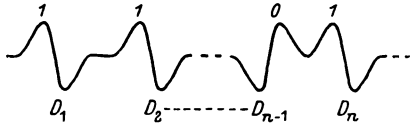


Figure 12.10 Induced voltages for magneto-acoustic storage

difference between pulse size at D_1 and at D_n. In some experiments lengths of 40 ft. have been employed and it has been found satisfactory to coil these on rubber sponges.

For completeness, it must be mentioned here that the magnetic-acoustic store can be used as a delay line store of the type described in a later section. In this application the output can be greatly increased by placing a polarizing magnet over the output coil D_1, say.

RECTANGULAR LOOP MATRIX STORAGE

Although it is true to say that no storage device, using macroscopic elements, would be acceptable as the main high speed memory of a modern automatic computer, in view of the want of any really satisfactory high speed store there is a growing body of opinion in favour of a relatively small ultra high speed memory backed up by a large medium speed memory.

In the field of ultra high speed the following matrix device, suggested by Forrester, has given promising results.

Two sets of wires $(a_1\ a_2\ a_3\ a_4)$ $(b_1\ b_2\ b_3\ b_4)$ (*Figure 12.11*) are arranged to form a two-dimensional grid. At the intersection of each pair $(a_r b_s)$ is placed a ring core C_{rs} of a rectangular loop

magnetic material of the type described in Chapter 10. The ring is so arranged that both a_r and b_s pass through it.

Suppose, now, that one of the a wires a_R, say, and one of the b wires b_S receive simultaneously current pulses sufficient to produce individually fluxes less than H_{min} (*Figure 12.12*) in any core which they thread. Under these circumstances all cores C_{Rs} C_{rS} $(r \neq s)$ will receive flux changes of H_{min} and these will not change their magnetic state. C_{RS}, however, will receive flux change 2 H_{min} and this will be sufficient to put it in to state 1 (*Figure 12.12*) if it was not already in that state.

To read out from the array several schemes are available according to the type of output required. For parallel output, if wire b_S

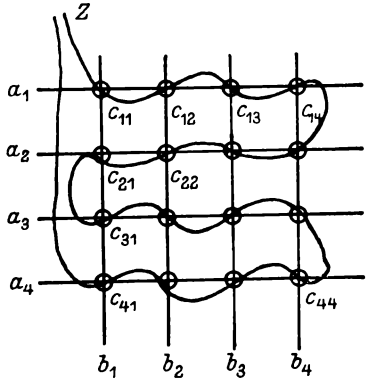


Figure 12.11 Matrix store

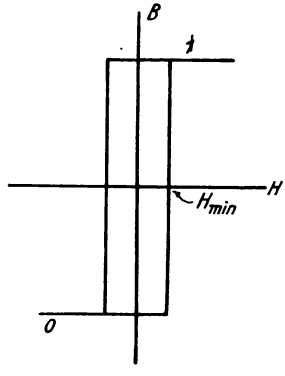


Figure 12.12 Rectangular loop characteristics

is pulsed with current sufficient to produce a flux greater than H_{min} in all cores which it threads, each element C_{rS} will emit an impulse to the a_r wires if previously magnetized to state 1, but not otherwise. If an individual output is required from C_{RS}, the whole array is threaded by an extra and continuous wire Z, upon pulsing a_R and b_S with currents adequate to produce a total flux in C_{RS} greater than H_{min}, an output will be produced in Z if, and only if, C_{RS} was initially in state 1.

Since only single wires thread each core large currents are required to produce adequate fields; these are very suitably obtained by means of the transformer type function table described in Chapter 11, *Figure 11.8.*

Unfortunately the act of reading out from a matrix of the type described, destroys the information stored in the elements concerned

and a recycling device has, in general, to be added. Even with this disadvantage, however, a 16×16 matrix has been operated, using ferramic storage elements, with a total read out plus regenerate time of only 10 μsec.

It is worthy of mention at this point, that using barium titanate as the dielectric, capacitors displaying saturation and remanence characteristics have been constructed; using these a matrix store, similar to that described above, can be constructed by painting lines of conducting material on either side of a thin dielectric sheet. Tests have shown that this device is potentially capable of speeds up to one order of magnitude faster than those attainable with the magnetic cores. Since the objection of macroscopic elements is largely removed in this essentially surface operating device, and furthermore, a function table for input to the matrix can be constructed on the same plan, it seems probable that really large high speed memories of this type will, in the future, be practical.

SONIC DELAY STORAGE

One of the first and most elegant high speed storage devices to be suggested was the sonic delay line. This consists, basically, of a sonic conductor *c* (*Figure 12.13*) supplied with circuits, *I* for transforming an incoming signal into a sonic pulse, receiver *O* capable of detecting the transmitted sound pulse and transforming it back into electrical form, and a gate *G* either for recycling the output from *O* to the input *I*, or for permitting the insertion of data from an external source.

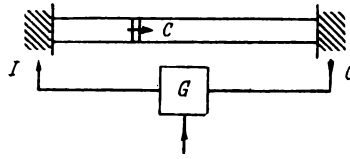


Figure 12.13 Sonic delay line

It is clear that if $n\triangle$ is the delay between insertion of the sonic pulse at *I* and its transmission and reception at *O*, *n* binary digits separated by time intervals $\triangle$ can be stored in the line.

In the simple form described the device would be unsuitable for incorporation in a calculating machine but a practical design is shown in *Figure 12.14*. Here the delay medium is usually mercury, this has the advantages of a relatively low velocity of propagation

(145 cm/msec), of good acoustic match with the quartz transducers used for input/output, and of freedom from the unwanted modes of sound transmission found in solid media.

Input and output are *via* quartz crystals Q_1 and Q_2 usually tuned to from 10 to 20 Mc/s. On the input side the crystal Q_1 is driven from an impedance matching network P and the output from Q_2 is amplified by means of a tuned amplifier A.

Since dispersion and attenuation occur in the delay line an input pulse of the type shown in *Figure 12.15* (*a*) appears, after transmission and amplification, blunted and broadened as shown in *Figure 12.15* (*b*). In consequence of this, data would be lost after

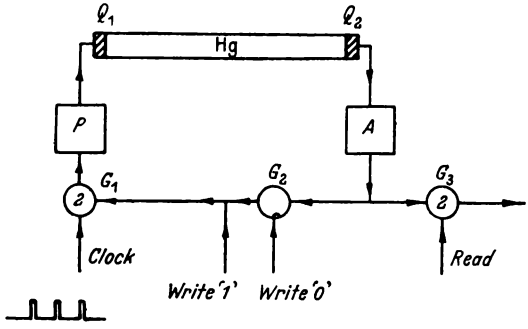


Figure 12.14 Delay line

one or two passages down the line if no reshaping were applied. Reshaping is performed by applying the amplified pulse to the input of a gate G_1 to the other input of which is applied a standard clock pulse shown in *Figure 12.15* (*c*). The gate output then has the form of the clock pulse and not that of the distorted input, so that the data in the line is regenerated at each cycle.

In order to obtain the output of a particular digit or group of digits from the line a gate G_3 is provided which can be opened at given times from an external source. In a like manner, to insert data into the line an inhibitory gate G_2 is provided. If '1' is desired in a given position this may be inserted directly to the input of G_1 where, if no '1' is already arriving from A and G_2 it will insert one, whereas if '1' is already arriving its effect will be merely redundant. Similarly G_2 is inhibited if the insertion of '0' is required.

Lines have been constructed of mercury, aqueous alcohol, quartz and magneto-strictively pulsed nickel wire. The usual length is

1,024 digits corresponding to a delay of from 1 to 2 msec and a physical length of from 1–2 m according to the inter-digit interval used. The chief source of trouble lies in temperature variation, since the velocity of propagation is sharply dependent upon this factor. Mercury delay lines are usually enclosed in a thermostatically controlled constant temperature container. It is important to ensure good acoustic contact between the quartz transducer and the mercury; this can be achieved by first filling the outer envelope with absolute alcohol and then displacing it with the mercury.

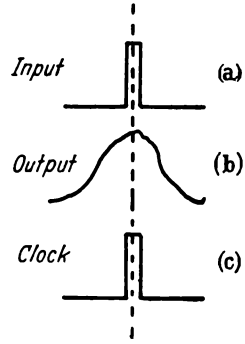


Figure 12.15 Input and output waveforms

On applying the end quartz crystal the residual alcohol is squeezed out and a good quartz–mercury contact remains.

Numerous devices have been suggested for the elimination of reflections in delay lines and also for their external adjustment for total delay, but as it seems likely that the magneto-strictive wire will eventually replace the mercury line and does not require these elaborations, it does not appear to be worth describing them here.

DIELECTRIC STORAGE

The observation that a condenser will hold a charge of electricity for a relatively long period, depending only upon the leakage characteristics of the system, suggests that dielectric media might be used as a computer storage device. In principle, at least, charge could be deposited upon the surface of a rotating dielectric drum and detected later in a similar manner to that used in the magnetic drum storage system.

This elementary storage system seems never to have received practical trial, but several others, depending on dielectric storage but operating by means of electron beams *in vacuo*, have been extensively developed.

Before entering into details it is necessary to consider possible means of rendering the storage permanent over any desired period of time, and two general methods suggest themselves. In the first the data stored at any point of the medium is examined and is then regenerated. In the second direct use is made of the secondary emission properties of the dielectric storage surface under electron bombardment.

Consider the physical situation pictured in *Figure 12.16*. An electron gun *g* generates an electron beam *b* which is focused

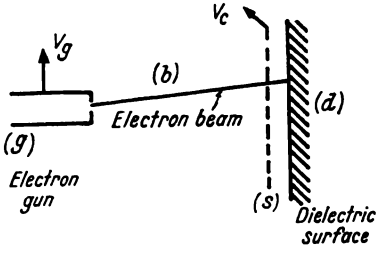


Figure 12.16 Schematic of dielectric storage device

upon a dielectric screen *d*. A mesh screen *s* is placed in front of *d*.

When primary electrons are incident upon a dielectric *in vacuo* several courses of action are possible:

(i) The surface may emit more electrons than it receives.
(ii) It may emit the same number.
(iii) It may emit fewer electrons than it receives.

In these circumstances it is said to have a secondary emission coefficient greater, equal or less than one.

With the physical situation of *Figure 12.16* the variation of secondary emission coefficient of *d* with potential of *d* is shown in *Figure 12.17*.

For low potential differences between *g* and *d* electrons striking *d* have low energies and eject few secondaries, corresponding to points between *l* and *m* on the curve. As the potential of *d* is increased the secondary emission coefficient rises with increasing incident electron

energy and becomes greater than unity at *m*. For further increase in the potential of *d* the coefficient first increases and then starts to drop owing to the ability of *d* to recapture the electrons liberated by the initial bombardment, this corresponds to the curve *m*, *n*, *o*. It has been assumed throughout that the grid *s* of *Figure 12.16* is able to capture and remove all 'free' electrons from the system.

It follows from the above qualitative argument that the system of *Figure 12.16* can exist in two stable states corresponding to *m* and *n* on the curve of *Figure 12.17*, thus all of the conditions for a binary storage device are present.

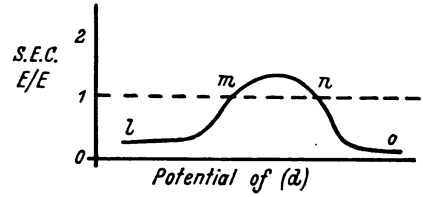


Figure 12.17 Secondary emission coefficient v. *potential of* (*d*)

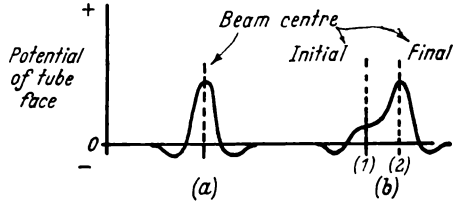


Figure 12.18 Potential distributions in Williams storage system

In the first of the storage systems to be described, due to F. C. Williams, a standard cathode ray tube is used with the addition of a collector plate on the external surface of the tube face.

If the beam is turned on to a spot of the tube face with such an energy as to ensure a secondary emission coefficient greater than unity and then turned off, the potential distribution has the form given by *Figure 12.18* (*a*). If the beam is now displaced slightly, the secondaries generated in the new position partially fill up the positive hole generated in the first operation, the resulting potential being then as shown in *Figure 12.18* (*b*).

Suppose, next, that the beam is turned on to the positions *a* or *b* (1). In the first case a large negative charging signal will be emitted

from the external collector plate, and in the second case a much smaller signal. It thus follows that the type of distribution present on the inner surface of the tube can be detected at the initial positions *a* or *b* (1) so that any data stored in such a system can be periodically examined and regenerated.

The system described above is usually known as 'dot-dash' recording, another method producing as good or better results is known as 'focus-defocus' and operates by first applying a finely focused beam to the recording spot and then defocusing it.

In the Williams system data is stored in serial form on the tube face, a television type raster of 32×32 spots being used. When read out is not required the circuits are so arranged that data is scanned row by row and regenerated continuously. Upon read out being requested the row under regeneration is completed after which the scanning beam jumps to the row required, reads it out,

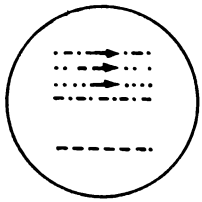


Figure 12.19 Williams scanning raster

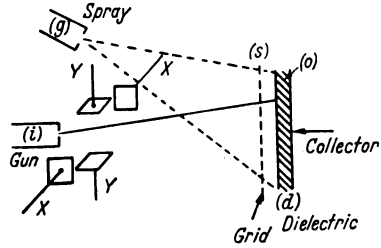


Figure 12.20 Forrester-Haeff storage system

and then proceeds with the regeneration process starting with the next row. The process of writing is exactly similar, except that new data is inserted into the selected line.

Apart from the raster generating and selecting circuits, the logical techniques of reading, writing and regeneration are practically identical with those described for the acoustic delay line, and shown in *Figure 12.14.*

A similar storage system, but operating in the parallel mode, has been constructed at the Institute for Advanced Study, Princeton; in this the beam performs a continuous regenerative scan as before but can be switched after any particular spot instead of at the end of a row.

A disadvantage peculiar to this system of storage is the so-called 'read-around' constant. By this is meant the fact that if one

particular row or spot is called for with more than a certain frequency the regeneration of part of the other data is inhibited and the latter lost. This phenomenon causes considerable loss of time over the optimum possible in its absence and also leads to complexities of coding not present in other systems.

It was hoped, in the early days of the Williams system, that ordinary commercial cathode ray tubes could be used. Owing to the presence of areas of defective secondary emission on such tubes, however, this has not proved possible and specially constructed tubes are now used.

The second type of dielectric storage system has been constructed by Haeff and by Forrester. This makes direct use of the two state

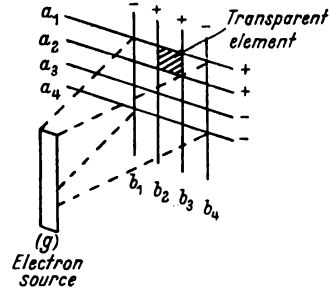


Figure 12.21 Selectron grid principle

property of the storage medium discussed above with reference to *Figure 12.17.*

The general layout of the system is shown in *Figure 12.20.* The gun *g* is so arranged as to 'spray' the whole of the dielectric surface with electrons, under these circumstances any binary charge distribution present will be maintained.

To read out data a second gun *i* generates a finely focused electron beam which can be directed to any point of the dielectric by means of deflector plates *x* and *y*. When this is done an instantaneous signal will be produced at the collector, whose magnitude will depend upon the potential of the dielectric at the point concerned.

Numerous methods of data insertion exist, as in the Williams system, and a typical method consists in the use of the *i* beam alone to record '1' and of the *i* beam and a voltage pulse on collector plate or grid to record '0.'

An advantage of the system lies in the fact that, as the dielectric and collector plate are normally built into the tube, a thin dielectric of predictable quality can be used and a considerably larger signal than that available from the Williams tube obtained. In addition, read-around troubles are largely eliminated in these systems.

A final example of dielectric storage is the 'Selectron,' the actual storage principle in the most recent model consists of macroscopic elements, the interesting feature, however, lies in the means provided for selecting the particular element required. Essentially the selector consists of two perpendicular grids of wires, $a_1 \ldots a_n$, $b_1 \ldots b_n$, of *Figure 12.21.* A source of electrons g generates an

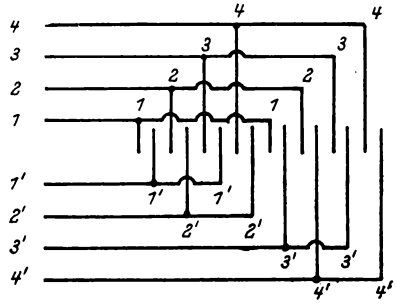


Figure 12.22 Selectron combination bars

electron cloud, this is attracted to the grids a, b by the action of those elements at positive potential. On attempting to penetrate the grid, however, the electrons are repelled by all regions except that bounded by four positive conductors; in this way a particular window is selected and the storage region behind it can be used for reading or writing in any of the methods previously described.

As described above the grid selection principle gives access to N^2 storage places with only $2N$ input wires. By means of a further refinement, however, this may be further improved as shown in *Figure 12.22.*

Consider one grid of 16 wires and number these as shown in *Figure 12.22,* it is readily seen that if one member of each array 1 2 3 4, $1^1\ 2^1\ 3^1\ 4^1$, is made positive and the rest negative, only one pair of adjacent wires will be positive. It follows that one window has been selected from 16 by means of 8 inputs.

In general, it can be shown that 2^{N+1} input wires suffice for the control of 2^{2N} adjacent positions so that, considering the two grids at right angles, $(N/4)^4$ windows are controlled by N input wires, N being of the form 2^m. Whilst this is not quite so efficient as a truly binary control system, it is considerably more efficient than any other scheme. It should be noted that the principle can be applied directly to the selection of matrix points in the rectangular loop storage systems previously described.

So far Selectron valves are available having a storage capacity of 256 binary digits and an access time of about 20 μsec, but there seems to be no reason why a larger version should not be developed in time.

THE SIZE OF AN ADEQUATE STORE

The answer can be given to this question in a trivial manner, by the one word 'infinite.' It is perhaps a more adequate question to ask 'what is the minimum storage acceptable in a high speed computer?'

In 1947, before any high speed computer possessing storage facilities had been actually completed, a survey made at Princeton had suggested that 4,096 words of about 10–12 decimal (i.e. 35–45 binary) digit accuracy might be appropriate as the high speed internal storage for a machine. This should, of course, be backed up by a slower speed storage on some such medium as magnetic wire or tape, capable of completely evacuating or replenishing the main store in from ·1 to 1 sec and having a capacity of 10^5–10^6 words.

Now that actual computers have been built and operated it appears that this estimated storage requirement, whilst desirable mathematically, is not at the moment (1952) a really practicable engineering proposition. A close examination of the work done by existing machines seems to show that a good practical size for the high speed store is 512–1,024 words and that a slower storage of 2×10^4 words is adequate to back it.

Considerable work has now been done on ultra high speed storage on ferro-electrics, and it appears likely that at least some future computers may have three distinct storage organs:

(1) An ultra high speed unit for 32–128 words with access time 1–5 μsec.

(2) A high speed main store for 1,024–4,096 words on magnetic drums or similar devices and having an access time of about 1 msec.

(3) A low speed store, on magnetic tape or wire able to refill or empty (2) in 1–·1 sec and having a capacity of 10^5–10^6 words.

These units would, of course, be backed up by arithmetic and control circuitry of adequate speed and form a part of a machine whose operation would almost certainly be in the parallel mode.

13

DEFINITION OF A CODE AND DISCUSSION OF ITS FORM AND CONTENTS

THE code of an automatic computer is simply the list of orders, arithmetic and otherwise, which it is possible to make the machine obey. The sequence of such orders appropriate to a calculation will generally be stored in the memory, together with any numerical data necessary and it is, therefore, convenient to express them in coded numerical form, whence the use of the word 'code' in this context. The set of operations necessary to solve any particular problem is also sometimes referred to as the code, but the less ambiguous term 'programme' is more generally used; the process of preparing the set of orders is known as coding or programming.

THE PHYSICAL FORM OF A CODE

Before attempting to define the basic operations necessary for a complete code it is convenient to discuss the physical form of the orders. The most important variation in this respect is in the number of memory locations (or addresses) referred to in each order; this may vary from one to four.

Thus a typical order from a one-address code might read:

> 'Add the number in memory position (x) into the accumulator.'

The set of such one-address orders forming a programme are normally stored in consecutive positions in the memory, the control of the machine being constructed to select the next order from the memory position following that in which the previous one was stored.

The corresponding order in a two-address code would read:

> 'Add the number in memory position (x) into the accumulator and then execute the order stored in position (y).'

The control in this type of machine takes its sequence from the programme itself.

A three-address code may specify the location of two numbers and the position of the next order thus:

> 'Add the number in memory position (x) to the number in memory position (y) and then execute the order stored in position (z).'

Alternatively, instead of specifying a second number the memory location in which the result of the operation is to be stored may be given. Several permutations of this sort are available and will be readily constructed by the reader.

Finally the four-address code combines all variants mentioned above in orders of the form:

> 'Add the number in memory position (w) to the number in memory position (x), send the result to memory position (y) and then execute the order stored in memory position (z).'

In discussing the relative merits of the various types of code four points must be considered. These are: (a) Simplicity in use. (b) Economy of memory space. (c) Overall speed of execution. (d) Complexity of circuitry.

From the point of view of simplicity the one-address code is undoubtably preferable since it is normally only necessary to specify the location of the first order in a programme, subsequent orders and parameters following in sequence. Of course, this is a somewhat simplified picture of the process of coding as will be realized when the subject of subroutines is discussed. Nevertheless the fact that the set of orders relevant to a particular calculation is stored in the memory *en bloc* and not scattered about does simplify the assignment of memory locations when coding. Moreover, as will be seen later, the time of execution of orders is normally taken into account when programming with a multi-address code, and this obviously complicates the process.

On the question of economy of memory space it might, at first sight, seem that the three- and four-address codes are necessarily superior since they embody two or three orders of the one or two-address type. It should be noted, however, that an order specifying, say, three memory locations will require more digits for its representation than one specifying only one, and, while it is usually possible to store *two* orders of the one (and possibly two) address type in *one* memory position, this is generally not possible with multi-address codes. Of course the use of half-length memory positions has the disadvantage that it is necessary to provide means for splitting up the contents of a memory position and operating first on one half and then on the other.

A final point may be mentioned in this connection. When using a one-address code it is sometimes necessary to shift the control from one set of operations to another not stored in sequence in the memory, and this is done by means of an order of the form: 'Transfer control to memory location (x).'

This order is, of course, redundant in a multi-address code since it is embodied in the code itself.

Before discussing the third point—overall speed of execution—a brief diversion must be made to define the two possible methods of selecting data from the memory. This may be done by means of a coding switch which enables connection to be made directly with any memory position as, for example, in the cathode ray tube storage developed by Williams. This method is logically preferable, but unfortunately considerable technical difficulties are encountered in making such a selector sufficiently fast and compact; for this reason most storage devices operate on the alternative scheme which may be called the scanning method. Here the contents of the memory are continually scanned by circulating them, either electronically as in the delay line, or mechanically as in the magnetic drum. The disadvantage of this system is that, on the average, it is necessary to wait for half a complete scan time before any particular piece of data is available, and this may reduce the overall speed of operation of the computer considerably. The advantage of the multi-address code is now apparent, for, whilst in using a one-address code, orders must be arranged strictly in sequence and with no reference to their availability when needed, the multi-address code makes possible the strategic disposition of data so that the least possible time is wasted in scanning the memory for the next order or number required. This process is known as optimum coding.

Thus, taking as an example the two-address order:

> 'Add contents of memory position (x) into the accumulator and then execute order stored in memory position (y).'

and supposing that this order is stored in memory location (1), the disposition of (x) and (y) might be as follows:

Memory location	Contents
1	Addition order
2	—
3 ($=x$)	Number to be added
4	—
5 ($=y$)	Next order

It has been assumed here that the time taken by the control to decode the order and the time taken to perform the addition are both less than the scanning time of one memory location. In most machines this is certainly true. Positions (2) and (4) would, of course, in general be filled by other portions of the programme.

It will be seen from this example that the time taken to interpret and execute the addition order, and to locate the next order is equal to the scanning time of four memory locations. The corresponding time using a one-address code would be approximately equal to the scan time of the whole cycle and it is therefore evident that the overall speed of this type of machine can be greatly increased by the use of a multi-address code.

Finally, on the question of circuit complexity, it can be said that a two-address code is easier to engineer than any other variety. This matter has been discussed in detail elsewhere and will therefore not be pursued further here.

THE CONTENTS OF A BASIC CODE

The choice of an order code requires careful consideration since on it depend, on the one hand, the engineering difficulties encountered in constructing the 'hardware' of the computer, and, on the other, the ease with which a problem can be programmed for the machine.

The four basic arithmetic operations are addition, subtraction, multiplication and division, and the machine must be capable of performing these either automatically or by programming. Of these, addition and subtraction are the more fundamental, since both multiplication and division can be programmed from them; it may be remarked also that, as Williams has pointed out, subtraction is more versatile than addition, since addition can be performed using a subtractor by means of the relation $+x \equiv -(-x)$, whereas if only an adder is available subtraction is impossible.

Even addition and subtraction, however, are not essential since it is possible to programme them on a machine consisting only of two shifting registers connected in series, a memory, facilities for transferring numbers between these units, and a means of stopping the machine. However desirable such a simple machine may be from the engineering point of view it is quite impractical mathematically, as will be seen when it is realized that about 30 orders would be needed to add two binary *digits* together with any carry over. It will therefore be assumed that the operations of addition and subtraction are available.

The decision must now be made as to whether multiplication and division are to be included in the order code, or whether they are to be programmed, and it is pertinent to examine more closely the form of programme required. Division, although again the more fundamental operation, since $xy \equiv x \div (1 \div y)$, will not be considered here as it has been pointed out in an earlier chapter that it requires a greater engineering complexity than multiplication. It can either be programmed directly by a method similar to that used for multiplication, or by an iterative method embodying the multiplication routine.

In order to examine the possibility of programmed multiplication it will be assumed that the machine contains an accumulator and a shifting register capable of connection in series so that the right shift operation transfers the extreme right-hand digit of the accumulator into the extreme left-hand position of the register. A one-address code will be used for simplicity of presentation, and it is therefore necessary to assume the existence of a control transfer order of the form discussed in the previous section. The following is a typical order code and notation.

Order	Explanation
$+M(x)$ to A	Add contents of memory position (x) into the accumulator (A)
$-M(x)$ to A	Subtract contents of memory position (x) from the accumulator
$r(n)$	Shift contents of the accumulator and register (R) n places to the right, transferring the terminal digits of (A) into the most significant end of (R)
$M(x)$ to cA	Transfer contents of $M(x)$ to the cleared accumulator
$M(x)$ to cR	Transfer contents of $M(x)$ to the cleared register
R to cA	Transfer contents of (R) to the cleared accumulator
C to $M(x)$	Transfer control so that the next order executed is that stored in $M(x)$
Stop	

THE PROGRAMME FOR MULTIPLICATION

Suppose now that (a) and (b) are the two n digit positive binary numbers to be multiplied. To form the product the following process is necessary.

(1) If $a_m = 1$ (1.1) Add (b) into partial product already formed in (A).

(1.2) Shift contents of A one place to right.

(2) or if $a_m = 0$ (2.1) Add partial product into *cleared* accumulator and shift one place to right.

This process is repeated, the digits of (a) being examined successively starting at the least significant. To formulate the process in terms of the code given above, suppose that the set of orders corresponding to (1) above starts at memory position p_2 (which corresponds to unity in the p_2th digit of the order) and the second set at memory position 0.

The programme is then as follows:

Memory position	Contents	Explanation
p_1	$+M(c)$ to cR	$M(c)$ contains zero and this clears R.
p_1+1	$+M(\bar{a})$ to cA	a_m is isolated and transferred to the accumulator, the remaining digits of (a) being stored for further use
p_1+2	$r(1)$	
p_1+3	A to $M(\bar{a})$	
p_1+4	R to cA	
p_1+5	$r(p_2-1)$	The order stored in $M(p_1+8)$ is adjusted by means of a_m, so that two different courses will be followed according as $a_m=0$ or 1
p_1+6	$+M(p_1+8)$ to A	
p_1+7	A to $M(p_1+8)$	
p_1+8	C to $\begin{cases} M(0) & a_m=0 \\ M(p_2) & a_m=1 \end{cases}$	
$a_m=0$		
0	$+M(p)$ to cA	The partial product (p) is shifted one place to the right in accordance with (2.1)
1	$r(1)$	
2	A to $M(p)$	
3	$M(c)$ to cR	R is cleared
4	$+M(I)$ to cA	This is a counting process to determine when n cycles of addition and shifting have been completed. See below
5	$r(1)$	
6	A to $M(I)$	
7	R to cA	
8	$r(p_1-1)$	The result of the counting process is used to modify the order in $M(11)$
9	$+M(11)$ to A	
10	A to $M(11)$	
11	C to $\begin{cases} M(p_1) & I \geqslant 2^{-n} \\ M(12) & I < 2^{-n} \end{cases}$	
12	Stop	

Memory position	Contents	Explanation
$a_m = 1$		
p_2	$+M(p)$ to cA	(b) is added to the partial product (p) and shifted
p_2+1	$+M(\bar{b})$ to A	
p_2+2	$r(1)$	
p_2+3	A to $M(p)$	The order contained in $M(p_1+8)$ is restored to its original state ready for the next cycle, $M(d)=2^{-p_2}$
p_2+4	$+M(p_1+8)$ to cA	
p_2+5	$-M(d)$ to A	
p_2+6	A to $M(p_1+8)$	
p_2+7	C to $M(3)$	
$\bar{a}$	Initially a	
$\bar{b}$	b	
c	0	
d	2^{-p_2}	(i.e. 1 in p_2th place, zeros elsewhere)
I	Initially 2^{-1}	
p	Partial product	

Orders 3 to 11 require a little more explanation. They form a counting process by means of which it is possible to determine when n cycles of addition and shifting have been completed. $M(I)$ initially contains 2^{-1}, and it will be seen that after n cycles this will have become '1' in the first stage of the register. The presence or absence of this digit in the register is then used to modify order (11) directing the control either to return to the beginning of the cycle or to stop the calculation.

A similar process is followed in orders p_1+3 to p_1+7 but this time the modifying digit is supplied by (a).

This programme is uneconomical in the extreme, requiring 36 memory locations for the multiplication of two positive numbers, and it is not suggested as a practical routine. It does, however, demonstrate the logical completeness of the abbreviated code outlined above. Such a code would only be justified in a machine of extremely high speed where the technical difficulty of providing more complex orders might be great.

As almost all machines at present completed or contemplated do contain facilities for automatic multiplication, this will be assumed hereafter in coded examples.

ITERATION AND THE CONDITIONAL TRANSFER ORDER

It was seen in the programme for multiplication given above that a counting process is necessary to determine when a certain number of cycles have been completed.

This introduces an important point in the use of high speed computers, namely that they are most economically employed in calculations involving the frequent repetition of a few processes. Thus it would be unwise to contemplate using a high speed computer for a single problem in which no repetition occurred, since the process of programming and preparing information for the machine would probably occupy at least as long as manual computation of the solution. Fortunately most problems occurring in practice can be reduced to these repetitive processes or iterations as they are usually called, and often a single problem will contain many such iterations.

It is evident, then, that counting will be an important factor in programming and it is very desirable to provide a more convenient means for accomplishing it than the rather clumsy method suggested above. This can be done by the introduction of a 'conditional control transfer' order which is usually of the form: 'If the number contained in the accumulator is $\geqslant 0$ transfer the control to the sequence of orders beginning in $M(x)$, but if it is < 0 continue in sequence (or in the case of a multi-address code transfer to $M(y)$). This will be abbreviated to:

$$Cc \text{ to } M(x)$$

There are slight variations in the form of this order in different computer codes, but the effect on programming is not significant and the above form will be used in subsequent programming.

Suppose, now, that it is desired to repeat a certain routine n times, and then to switch the control to another sequence of orders. An iteration index which will initially have the value $2^{-m}(n-1)$ (where m is the index of the least significant digit handled by the computer) is stored in the memory. After each repetition of the routine 2^{-m} is subtracted from it and the Cc order used to discriminate on the remainder. Until the nth repetition this will be $\geqslant 0$ and the control will be directed back to the beginning of the cycle again, but after the nth repetition it will become negative and the control will then move to the next section of the programme.

One other important use of this order must be mentioned; this occurs when a process of successive approximation is used to calculate a quantity—the roots of a polynomial or the square root of a number, for example. In calculations of this kind it is sometimes difficult to determine beforehand how many steps of the approximation will be necessary, and the use of an iteration index as described above is therefore impossible. A much better method is available, however. for, suppose that x_n and x_{n+1} are two successive values of

the approximation, and that $x_{n+1} > x_n$. If, after each step the value of $x_n - x_{n+1}$ is calculated, the result will be negative until $x_n = x_{n+1}$ and the *Cc* can be used to determine when this occurs. When this point is reached, of course, the approximation has proceeded as far as possible within the range of accuracy of the machine, and the control is transferred to the next part of the programme.

PRACTICAL CODES

The basic operations discussed above are essential in any practical code, but the introduction of other orders depends on the individual machine. In particular, the engineering difficulties likely to be encountered, the importance of reducing overall operating time and of economizing in the use of memory space by reducing the number of orders needed for a calculation must be considered. Thus, the inclusion of a division order, although not essential, is certainly desirable; the argument in favour of a square-root order is less strong, however, since it is an operation of much less frequent occurrence in general computing. Most codes also include certain non-arithmetical operations concerned with transferring numbers, or portions of numbers, from one part of the machine to another. These are introduced for convenience in coding and their form varies with the particular machine concerned.

It is now proposed to give a selection of codes and orders evolved for use with specific machines as this is the most convenient means of illustrating some of the desirable features of a practical code. The first example will be the single-address code proposed for use with the machine at the Institute for Advanced Study, Princeton, where much of the pioneer work on coding was done under the direction of von Neumann.

	Symbol	Explanation
1	*x*	Clear accumulator and add number from memory location *x* into it
2	*x* –	Clear accumulator and subtract location *x* into it
3	*xM*	Clear accumulator and add absolute value of number in location *x* into it
4	*x* – *M*	Clear accumulator and subtract absolute value of location *x* into it
5	*xh*	Add number in location *x* into accumulator
6	*xh* –	Subtract number in location *x* into accumulator
7	*xhM*	Add absolute value of location *x* into accumulator
8	*xh* – *M*	Subtract absolute value of location *x* into accumulator

	Symbol	Explanation
9	xR	Clear register; transfer number in location x into it
10	A	Clear accumulator and shift number in register into it
11	$x\times$	Clear accumulator and multiply the number in location x by the number in the register. The left-hand 39 digits are placed in the accumulator and the right-hand 39 digits in the register. The sign digit of the register is made equal to the extreme left (non-sign) digit. If the latter is 1, 2^{-39} is added to the accumulator
12	$x\div$	Clear register and divide the number in the accumulator by the number in location x. The quotient is placed in the register and the remainder left in the accumulator
13	xC	Shift the control to the left-hand order of the order-pair in location x
14	xC'	Shift the control to the right-hand order in location x
15	xCc	If the number in the accumulator is $\geqslant 0$ shift control as in xC
16	xCc'	If the number in the accumulator is $\geqslant 0$ shift control as in xC'
17	xS	Transfer the number in accumulator to memory location x
18	xSp	Replace the left-hand 12 digits of the left-hand order in location x by the 12 digits 9–20 in the accumulator
19	xSp'	Replace the left-hand 12 digits of the right-hand order in location x by the 12 digits 29–40 in the accumulator
20	R	Replace the contents $n_0n_1n_2\ldots n_{38}n_{39}$ of the accumulator by $n_0n_0n_1\ldots n_{38}$
21	L	Replace the contents $n_0n_1n_2\ldots n_{38}n_{39}$, $m_0m_1m_2\ldots m_{38}m_{39}$ of the accumulator and register by $n_0n_2n_3\ldots n_{39}0$ and $m_1m_2m_3\ldots m_{39}n_1$

It will be seen that automatic multiplication and division are both included, and that facilities are provided for working with the absolute value of numbers. A word of explanation is needed on the two control shift orders 13 and 14. It is intended in this machine to follow the method, mentioned earlier, of storing two orders in one memory location, and these orders provide a means for directing the control to either order of such a pair. For the same reason two conditional control transfer orders are provided. These

are not both strictly necessary, as one could be programmed from the other by suitable shift operations, but they will obviously facilitate the coding of problems.

The application of orders 18 and 19 is also not immediately obvious. One use occurs when it is necessary to refer to a tabulated function stored in the memory. Often the value of the argument will be calculated by the machine itself, and it will therefore not be possible to specify beforehand the memory location of the appropriate entry in the table. The details of the method used will be described later in the section on the use of tables and interpolation, but for the moment it is sufficient to say that the order directing the control to abstract a number from the table is included in the programme, but the portion denoting the memory position of the entry is initially left blank. When the argument has been calculated the appropriate memory location is inserted by means of orders 18 or 19.

Another important use of these orders is that they can modify a routine which has been coded in general form without reference to specific memory positions. This will be discussed in greater detail, however, in the section on subroutines.

Finally, a word on the shifting orders 20 and 21, which enable numbers to be shifted a single place to the left or right at each order. Some machines contain multi-shift orders of the form $L(n)$, $R(n)$ and these are to be preferred since, as will be seen later, multiple shifts are sometimes needed. The exact form of shift should also be noted. The Princeton machine follows the usual convention of working with binary numbers less than one, and of representing negative numbers by their complements modulo 2. Care is therefore needed when shifting numbers, that the sign digit is not lost when shifting left, and that negative numbers retain their complementary form when shifted right. It will be seen that both these conditions are fulfilled.

THE CODE OF A.P.E.(X.)C.

The code will now be given for *A.P.E.*(*X.*)*C.* (All Purpose Electronic X-ray Computer, Birkbeck College, London). This is included because it will be used in coded examples given later.

CODE OF A.P.E.(X.)C.

The address (y) in the following table refers to the location of the next instruction, so that the code forms a typical two-address system.

Order No.	Notation	Abbreviation	Description
1	$A(1-n)\rightarrow M(1-n)(x)(y)$	$A(1-n)\rightarrow(x)(y)$	Record first n digits of accumulator contents in first n positions of memory location (x). Go to (y) for next order •
		$A\rightarrow(x)(y)$	Form of (1) when $n=32$, i.e. whole word recorded in $M(x)$
2	$+M(x)\rightarrow cA(y)$	$+(x)\rightarrow cA(y)$	Add contents of $M(x)$ into cleared accumulator, *etc.*
3	$-M(x)\rightarrow cA(y)$	$-(x)\rightarrow cA(y)$	Subtract contents of $M(x)$ from cleared accumulator, *etc.*
4	$+M(x)\rightarrow A(y)$	$+(x)\rightarrow A(y)$	Add contents of $M(x)$ into accumulator, *etc.*
5	$-M(x)\rightarrow A(y)$	$-(x)\rightarrow A(y)$	Subtract contents of $M(x)$ from accumulator, *etc.*
6	$M(x)\rightarrow cR(y)$	$(x)\rightarrow cR(y)$	Transfer number in $M(x)$ to register, *etc.*
7	$M(x)\times R\rightarrow cA(y)$	$(x)\times R\rightarrow cA(y)$	Multiply number in $M(x)$ by number in register, first half of product in accumulator and second half in register
8	$r(n)(y)$	$r(n)(y)$	Right shift n places. Sign digit of accumulator propagates to right, 32nd stage of accumulator transfers to 1st stage of register. Digits are lost from right - hand end of register

Order No.	Notation	Abbreviation	Description
9	$l(n)(y)$	$l(n)(y)$	Left shift. Register and accumulator form cyclic system and no digits are lost
10	$R\rightarrow M(x)(y)$	$R\rightarrow(x)(y)$	Record contents of register in $M(x)$. After this operation register digits are all 0 or 1 according to initial state of $R(1)$
11	$Cc(x)(y)$	$Cc\geqslant 0(x)<0(y)$	If $A(1)$ is zero go to order in position $M(x)$, if $A(1)$ is unity go to order in position $M(y)$
12	Print (y)	$P(y)$	Print contents of accumulator. (Form of printing depends on external connections, i.e. Teletype or Hollerith)
13	$C(d+x)$	$C(d+x)$	Switch control to order located in position (x) on drum (d)
14	Stop	Stop	

Several unusual features of this code require explanation.

It will be observed that there are no orders concerned with reading to or from an external store (such as punched tape). These are unnecessary because it is proposed to store subroutines and tables in the very large, permanent magnetic memory with which *A.P.E.(X.)C.* will eventually be equipped. Any data and orders peculiar to a particular problem will be inserted directly from a teletypewriter, or from punched cards, and an input circuit is provided for this.

It is, of course, necessary to provide means for transferring from one drum to another and three distinct cases may arise. These are as follows.

(i) While following a programme of orders on drum (x) it may be necessary to insert a number into a subroutine on drum (y) and return to drum (x) for the next order.

(ii) While following a programme on drum (x) it may be necessary to read a number from drum (y) and return to drum (x) for the next order.

(iii) It may be necessary to transfer the control from a sequence of orders on drum (x) to a sequence on drum (y).

The only problem here is that space must be provided in the appropriate orders for the number (x, y, *etc.*) designating the drum under consideration. As will be seen later, only four digits are unallocated for other purposes, and, using these, access to $2^4 = 16$ drums would be possible; although this will suffice for some time it was considered advisible to make provision for the use of more drums if necessary. This is accomplished by using the six digits allocated to 'n' in orders $r(n)$, $l(n)$, $A(n)$ to M, to specify additional drums. It is thereby possible to obtain access to $2^{10} = 1{,}024$ drums and this will be amply adequate for any problem so far considered.

Thus, with the exception of the three orders mentioned, when a number is specified in these 10 digits, the appropriate drum will be selected before executing the order. The control will, however, return to the original drum for the next order in the sequence. If no number is specified the appropriate location on the current drum will be selected. It will be seen that, provided the order R to M is used when transferring to the memory, the first two contingencies mentioned above are provided for.

In order to obtain absolute transfer of the control from one drum to another it is necessary to provide a special 'bridging' order and order 13 fulfils this requirement.

Other interesting points are that multiple left and right shift operations are provided, and that the A to M order has been generalized to include also orders of the type A_L to M; when the whole contents of the accumulator are to be transferred, n is, of course, equal to 32.

Finally, it may be remarked that no 'modulus' operations are provided as in the Princeton code. The reason for this is that *A.P.E.(X.)C.* is a serial machine and the Princeton computer a parallel one. While complementing on a parallel array of digits is fairly simple, the same operation on digits appearing serially is by no means easy. Moreover, experience in coding has led to the conclusion that the modulus operations are comparatively rarely

required, and it was therefore decided not to include them in the code of *A.P.E.(X.)C.*

SOME SPECIAL ORDERS

Before concluding this survey of codes mention must be made of the so-called 'logical' orders which are built into the machines at the National Physical Laboratory and at Manchester University. These are as follows:

(1) A and $B=C$. A number C is generated whose digits C_n are one if both A_n and B_n are one and zero otherwise.

(2) $A \vee B=C$. C_n is zero if A_n and B_n are both zero, but is one otherwise.

(3) $\sim A=C$. $C_n=1-A_n$ (C_n is therefore the digital complement of A_n).

(4) $A \not= B=C$. $C_n=1$ if $A_n \neq B_n$, $C_n=0$ if $A_n=B_n$.

The last of these, used in conjunction with the conditional transfer order, enables the equality, or otherwise, of two numbers to decide which of two courses the calculation shall follow. Similarly, the order A and $B=C$ can be used to discriminate on any digit of a number. It can also be used to isolate a particular group of digits from a number by comparing the number with another containing unity in each position of the digits required, and zero elsewhere.

Another type of order proposed for the full-scale model of *A.C.E.* is known as an 'immediate instruction.' Using this, one order can initiate the addition of a series of pairs of numbers stored consecutively in the memory; this is of particular use in vector addition. Similarly the total of a series of consecutive numbers can be formed using only one instruction. It is also possible to perform operations on numbers containing more than the standard number of digits.

14

THE TECHNIQUE OF CODING

THE coding of a problem can be divided into three parts:

(a) Mathematical formulation.
(b) Schematic programme.
(c) Detailed programme.

The first of these will not be considered further except to remark that the methods used must involve only arithmetic operations, so that, for example, a differential coefficient must be replaced by its finite difference approximation. An investigation into the effect of such approximations, and of rounding off errors, on the accuracy of the final result is also necessary; furthermore, all quantities must be adjusted so that they do not grow or diminish beyond the range of the machine. The latter can be quite a serious problem, since it is by no means always possible to foresee the magnitude of numbers derived in the course of a computation, and this matter will be discussed later. The difficulty can, of course, be avoided if the machine is constructed to work with a 'floating' binary or decimal point; that is, each number has associated with it an index to indicate the position of the point, and the arithmetic unit of the machine takes account of these when operating upon the numbers. Such machines are, however, very difficult to engineer.

The complexity of the schematic programme depends chiefly on the number of iterative processes involved in the computation. The method of flow diagrams introduced by Goldstine and von Neumann is sometimes useful at this stage of coding and it will now be described briefly.

THE METHOD OF FLOW DIAGRAMS

The most straightforward type of calculation is that which involves no iteration or decision on the part of the machine; the programme then duplicates those operations which a human computer would follow, and the coding process can be represented by a single line as shown in *Figure 14.1.*

Suppose, for example, that it is desired to calculate the expression:

$$y = \frac{ax^2 + bx + c}{dx + e} \qquad \ldots . \quad (1)$$

for one set of values of x, a, b, c, *etc.* The sequence of operations will be as shown in *Figure 14.2.*

Figure 14.1

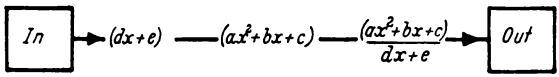


Figure 14.2

As mentioned before, this type of operation is unsuited to a high speed computer, since the time required for formulating the programme and feeding it into the machine will probably be at least equal to that required for manual computation of the problem.

A much more realistic problem would be the calculation of (1) for n values of x at equal intervals Δx. The process would then be to count the number of terms calculated by subtracting 1 from an iteration index I (initially equal to $n - 1$)—and after the calculation of each term apply the conditional transfer order to decide when the required number is reached. This embodies a 'simple induction loop' and is represented diagramatically in *Figure 14.3.*

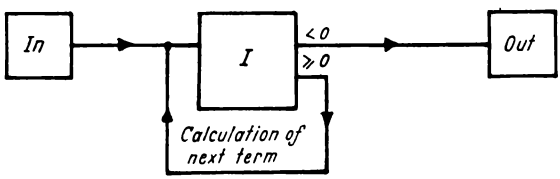


Figure 14.3

Now suppose that automatic division is not available and that it is necessary to evaluate $A_n = \dfrac{1}{dx + e}$ by an iterative process in which $A_n \leqslant A_{n+1}$.

Once again the conditional transfer order will be used, this time to decide from the difference of successive approximations $(A_n - A_{n+1})$ when the iteration has proceeded far enough. The flow diagram is shown in *Figure 14.4.*

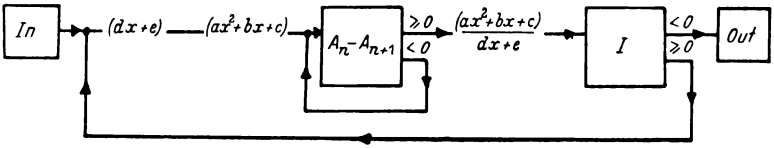


Figure 14.4

This is an example of a 'multiple induction loop.' More complicated patterns will be encountered in practice, of course, and examples and details of a scheme of notation to deal with these can be found in the original report on the subject.

THE DETAILED PROGRAMME

When the flow diagram is complete the next stage is to write out the detailed programme which will contain:

(a) A full list of all orders.
(b) Detailed memory locations for orders.
(c) Detailed memory locations for numbers and for transient storage required during the computation.

It is naturally impossible to lay down inviolable rules of procedure for coding; methods will vary with the type of problem in hand, the characteristics of the machine to be used, and the experience of the programmer. There are, however, many problems which have to be faced whatever method is used, and some of these are discussed below.

The detailed process of programming is best illustrated by examples, and some techniques which have been evolved for use with existing machines will be apparent from those given below. These techniques are not regarded as permanent, but may be varied with the problem under consideration, and with increased experience.

THE EXACT FORM OF ORDERS USED IN PROGRAMMES

As mentioned earlier, most of the programming will be in terms of the two-address code of *A.P.E.(X.)C.*, and before using this code it is necessary to explain in more detail the form in which orders are stored in this machine.

In common with most automatic computers *A.P.E.(X.)C.* operates in the binary scale with a fixed binary point, and normally handles numbers which are less than unity. These numbers consist of 31 digits together with a sign digit situated at the most significant end of the number; this is 0 for positive numbers and 1 for negative numbers, the latter being represented by their complement.

The code of *A.P.E.(X.)C.* contains less than 17 orders, and since these are to be represented in coded binary form, 4 digits will be required to specify any order. At the moment storage space is restricted to 512 locations, but the form of the memory is such that extra 'units' of storage for 512 numbers can be provided with little additional equipment. The storage locations of any particular memory unit are numbered from 0 to 511, and nine binary digits will suffice to specify any location. As the code is two-address, two such groups will be required.

The exact form of a coded order is as follows:

Digit	Contents
1–9	(x) Location of number to be operated upon, or location to which number is to be sent
10–15	(n) Counter
16–17	—
18–21	(c) Order code
22–30	(y) Location of next order
31–32	—

Locations 10–15 are used to specify 'n' in orders (1), (8) and (9). They are also used, together with locations 16, 17, 31 and 32 to specify the drum number as described earlier.

It should be noted that when storing numbers, as distinct from orders, the sign digit is located in position (1), and position (9), for example, is numerically equal to 2^{-8}.

As mentioned earlier, the location of the next instruction to be followed will be indicated in brackets after the order.

Thus:

$$M(6) \rightarrow cR(7) \text{ or } (6) \rightarrow cR(7)$$

means:

Transfer the number in memory location (6) to the cleared register, and then execute the order stored in memory location (7).

When the $R \rightarrow M$ order is used to record a number in a location on another drum (d), say, the order will be written:

$$R \rightarrow M(d+x)(y) \text{ or } R \rightarrow (d+x)(y)$$

which means:

Record the number in the register in $M(x)$ of drum number (d), and then execute the order stored in $M(y)$ *on the original drum.* A similar notation is used when reading a number from another drum thus:

$$+M(x+d)\rightarrow cA(y) \text{ or } +(x+d)\rightarrow cA(y)$$

Finally, all examples will be coded so that the first order appears in an arbitrary memory location denoted by m_n, where n varies from one problem to another. For clarity it will be assumed in most examples that orders are stored in consecutive positions m_n, m_n+1, m_n+2, *etc.*, and once specified for a programme, m_n will normally be omitted when referring to memory locations within that programme.

CODED PROBLEMS

It is now possible to consider more closely the actual process of coding, and as a first example the problem used to illustrate the method of flow diagrams will be coded in detail. For simplicity it will initially be assumed that a division order of the form:

$$M(x) \div R \text{ to } cA(y) \text{ or } (x) \div R\rightarrow cA(y)$$

is available. Later, however, the full problem will be coded.

It is required to calculate the expression:

$$y=\frac{ax^2+bx+c}{dx+e}$$

for values of x ranging from 0 to $n\triangle$ in steps of $\triangle$, a, b, c, d, e and $\triangle$ being given parameters. It will be assumed that the $(n+1)$ calculated values of y are to be stored in the memory, and it will therefore be necessary, after the calculation of each term, to modify the order dealing with this storage so that values are sent to consecutive memory positions. As will be seen from the previous paragraph, this can be done by adding 2^{-8} into the order after each iteration. It is also necessary to store an iteration index I (initially equal to $(n)\times 2^{-31}$); by decreasing this by 2^{-31} after the calculation of each term it will be possible to detect the completion of the required number of terms.

Storage locations must therefore be provided for the parameters a, b, c, d, e and $\triangle$ and also for 2^{-8}, 2^{-31}, I and the current value of x. These will be situated at the end of the programme, but, since it is not initially known how much space this will occupy, provisional locations will be allotted at first. It will, therefore, be assumed that

these quantities are stored in locations $M(k)$ to $M(k+9)$ respectively, and that the results of the calculation are to be sent to locations $M(l)$ to $M(l+n)$. It will also be assumed that the first order is stored in memory location (m_1). The programme is then as follows:

Tabulation of $ax^2+bx+c/dx+e$

Add m_1 to all memory locations.

Location	Contents	Remarks
0	$(k+3)\rightarrow cR(1)$	d
1	$(k+9)\times R\rightarrow cA(2)$	dx ($x=0$ initially)
2	$+(k+4)\rightarrow A(3)$	$dx+e$
3	$A\rightarrow(k+10)(4)$	$dx+e$ placed in temporary storage
4	$(k)\rightarrow cR(5)$	a
5	$(k+9)\times R\rightarrow cA(6)$	ax
6	$+(k+1)\rightarrow A(7)$	$ax+b$
7	$r(32)(8)$	Contents of accumulator transferred to register
8	$(k+9)\times R\rightarrow cA(9)$	ax^2+bx
9	$+(k+2)\rightarrow A(10)$	ax^2+bx+c
10	$A\div(k+10)\rightarrow cR(11)$	$ax^2+bx+c/dx+e$
11	$R\rightarrow(l)(12)$	Term stored
12	$+(k+8)\rightarrow cA(13)$	I
13	$-(k+7)\rightarrow A(14)$	$I-2^{-31}$
14	$Cc\geqslant 0(15)<0(\ \ )$	Index tested to see if programme complete
15	$A\rightarrow(k+8)(16)$	$I-2^{-31}$ substituted for I ready for next cycle
16	$+(k+9)\rightarrow cA(17)$	x
17	$+(k+5)\rightarrow A(18)$	$x+\triangle$
18	$A\rightarrow(k+9)(19)$	New value of x stored
19	$+(11)\rightarrow cA(20)$	Order disposing of y.
20	$+(k+6)\rightarrow A(21)$	2^{-8} is added to y disposal location, thus causing next location $(l+1)$ to be used in subsequent cycle
21	$A\rightarrow(11)(0)$	Order 11 re-inserted and control directed to start of cycle again

k	a
$k+1$	b
$k+2$	c
$k+3$	d

Location	Contents
$k+4$	e
$k+5$	Δ
$k+6$	2^{-8}
$k+7$	2^{-31}
$k+8$	I
$k+9$	x
$k+10$	Temporary storage for $(dx+e)$
l to $(l+n)$	Tabular values of $y=ax^2+bx+c/dx+e$

The 'next order' location, left blank in order (m_1+14), will depend on the rest of the programme. If this were the end of the calculation it would refer to a location containing the 'stop' order.

Since it is now apparent that 22 orders are needed, k takes the value (m_1+22) and the memory locations of a, b, c, *etc.* are given accordingly. $M(k+10)\equiv M(m_1+32)$ must be left vacant as it is used as a temporary storage, and the table of results can therefore be stored in memory locations (m_1+33) onwards.

It will be observed that in this example orders have been stored consecutively in the memory, and parameters and constants placed *en bloc* at the end of the orders. Although this arrangement is quite legitimate, it would not normally be used in a two-address machine, such as *A.P.E.(X.)C.*, since a higher overall operating speed is obtainable by the use of optimum coding as mentioned earlier.

This and the following chapter are intended as a general illustration of coding, however, and optimum coding will therefore not be used in the majority of examples as the general plan of the code is liable to be obscured when orders and numbers are scattered round the memory, as they must be when it is used. It should be emphasized, however, that the codes given could be used directly in a two-address machine although they do not represent the best possible arrangement for that particular type of machine.

15

THE USE OF SUBROUTINES IN CODING

THE coded example of the previous chapter was given as an introduction to the process of coding; it also serves, however, to illustrate one of the most important problems occurring in programming. It will be seen that the programme as given is only valid if stored in the stated memory positions and with the given parameters. In general such a routine would form only a part of the whole calculation, and it is, indeed, usually possible to divide large problems into a series of such sections, or subroutines as they are called, these being joined by interlinking orders. This use of subroutines is of great importance in coding, since it reduces both the labour required to programme a problem, and the chance of errors occurring in the programme. Thus the programme for calculating a square root, for example, can be coded, checked, and proved accurate by trial and thereafter stored, either magnetically or on punched tape, and used whenever required.

THE MODIFICATION OF SUBROUTINES

It will be seen that subroutines of the type discussed above require certain modifications before they can be used as part of a larger programme, and it is important that a convenient routine for doing this should be available. Some of the methods suggested in this connection will now be briefly discussed; the procedure evolved for use with *A.P.E.*(*X.*)*C.*, and which will be used in subsequent coded examples, will then be described.

It may first be observed that in general the necessary modifications are of two sorts. First the orders referring to memory locations within the subroutine must be changed according to the location of the subroutine in the memory, and secondly, the parameters relevant to the problem in hand must be inserted in the subroutine.

It is convenient to code subroutines on the assumption that the first order is always located in a fixed memory position and the first modification will then consist merely of adding a constant to all orders referring to memory locations within the subroutine. If

the fixed memory position is chosen as (0) this constant is equal to the number specifying the memory location in which the first order of the subroutine is to be stored. Thus, in the example given, if the first order were to be stored in $M(23)$, the addition of 23×2^{-8} (expressed in binary scale) would effect the necessary correction.

Since this is a purely arithmetic operation the machine itself can make these modifications and both of the methods now to be described make use of this fact.

THE PRINCETON MODIFICATION OF SUBROUTINES

The method now to be described is that suggested by Goldstine and von Neumann for use in conjunction with the Princeton computer.

It is based on the fact that normally all orders needing modification will necessarily refer to memory locations within the subroutine itself. If the subroutine is coded with its first order in $M(0)$, the memory location specified in such orders will therefore be numerically less than the total number of storage locations occupied by the routine. Thus, in the example given on p. 156, suppose that 10 terms are to be calculated, and that $m_1 = 0$. The total number of locations occupied would then be 42, and it will be seen that no order refers to a location higher than 42.

Two exceptions to this rule must be noted. First, there are orders of the type 'left shift' and 'print' which do not refer to memory locations and therefore require no modification. They will not, however, be invalidated if they are modified together with the rest of the programme, since the digits used for storing the memory location are never referred to, and this is, in fact, done.

The second exception is provided by those orders which refer to memory locations outside the subroutine. In practice these are almost always occasioned by reference to a tabulated function stored separately in the memory, and the memory location of the particular entry required will be determined by the calculations of the subroutine itself, and will therefore be unknown initially. In general, however, the memory location of the entry corresponding to a given argument will be equal to the sum of the argument and a constant which depends on the particular function in use. (See p. 180 for a fuller discussion of this point.) The modification of these orders is therefore similar to that described above except that a different constant is added. Consequently it is not possible to deal with them together with the rest of the subroutine, and they must be isolated and treated separately.

The isolation of such orders is effected by storing in them a dummy memory location, numerically greater than the number of positions occupied by the subroutine. This will eventually be replaced, but meanwhile it can be used to distinguish such orders by the following process. The physical arrangement of orders used in the Princeton machine is such that the memory location is stored in the most significant digits (excluding the sign digit). It will therefore be seen that, if the number of orders in the subroutine (multiplied by a suitable power of 2) is subtracted from each order of the routine in turn, a negative result will be obtained from each order requiring normal modification, but a positive result from orders of the type discussed above. Use of the *Cc* order then enables these to be treated separately. Of course, if the subroutine contains no orders of this type it can be modified by a simpler correction process which treats all the orders indiscriminately.

It may be remarked that, although this method was evolved for use with a one-address code, it can be adapted to a multi-address machine with the difference that *all* memory location specifications must be modified and with the limitation that optimum coding cannot, in general, be used as orders and parameters of a subroutine must be stored *en bloc*.

The details of coding for this method are rather lengthy, and will not be given here. They, together with an outline of the method of dealing with parameters, may be obtained from the original report on the subject.

THE E.D.S.A.C. MODIFICATION OF SUBROUTINES

The scheme in use with the *E.D.S.A.C.* is described by Wilkes, Wheeler and Gill. Briefly, the machine is provided with a set of initial orders which are permanently wired on to a uniselector. When the machine is switched on, these orders are automatically fed into the memory; they are so arranged that the necessary modifications are then performed on data fed in *via* punched tape, before the data is transferred to its final position in the memory. It is, of course, necessary to give an indication on the tape, of the orders and parameters needing modification, and this is done by means of a series of code letters and symbols, one of which follows each order to be modified on the punched tape. In addition there are several 'control combinations' which are, in effect, orders indicating the position in the memory to which the subroutine is to be sent; these must be specified anew each time the subroutine is used. These 'control combinations' and code letters direct the action of the input

orders and automatically switch the control of the machine to the first order of the programme when the modifications are complete.

This method has the disadvantage that the initial coding of subroutines is rendered more complicated by the necessity of including these code letters. Moreover, since the majority of orders in any subroutine will require modification, the presence of the code letters will appreciably increase the space occupied by the subroutine on punched tape. On the other hand, it may be argued that the coding of a subroutine must only be done once, and paper tape is cheap; these limitations are not, therefore, particularly serious.

STORAGE OF SUBROUTINES IN A.P.E.(X.)C.

The methods described above were designed for machines using 'non-permanent' high-speed storage of comparatively limited capacity (about 500 words). By contrast the magnetic storage of *A.P.E.(X.)C.* is permanent (magnetically recorded data has survived unchanged for 50 years) and the memory itself is of such a form that practically unlimited storage capacity can be contemplated. It is therefore possible to approach the problem from a different angle, and to consider storing subroutines permanently in the memory itself. This method has obvious attractions since it eliminates the necessity for most of the modifications described above. Moreover, the time of input of programmes will be considerably reduced because the only data to be inserted will be orders linking the subroutines, parameters and the orders necessary to locate them (these will be discussed in detail below) and any special routines which are not already included in the store. It may also be observed that magnetic storage of data is far more compact than punched tape or cards, and the method, therefore, is to be preferred for reasons of economy of space.

Of course a written record of routines must be kept in case they fade or become accidentally erased from the memory, but so far no trouble has been experienced on this score.

THE SUBSTITUTION OF PARAMETERS INTO SUBROUTINES

When subroutines are permanently stored in fixed memory locations only minor modifications are needed to certain of the orders of a routine before it can be incorporated in a programme, and the method of accomplishing these will be indicated on p. 164. There still remains, however, the problem of inserting the parameters

appropriate to a particular calculation and this will now be examined in more detail.

The parameters occurring in a subroutine can be divided into two classes—those which are known at the time the calculation is planned, and those which are derived in the course of the calculation. Thus, in the example on p. 156, n and $\triangle$ might be predetermined, and a, b, c, d and e the result of earlier calculation in the same programme.

Predetermined parameters can be inserted directly into the subroutine when the rest of the programme is fed into the machine. Care must be taken, however, when a subroutine is used more than once in the same programme, that any parameters which have been 'spoiled' by the use of the subroutine are restored to their original state. Thus, it will be seen from p. 156 that after the use of this subroutine the value of the iteration index I is no longer correct, and must be restored before the routine can be used again. Moreover, order $(m_1 + 11)$ must be modified so that the memory location for the storage of the first term is restored to its original value. In some subroutines it may also be necessary to clear out certain memory locations which are used for the storage of numbers and which must be empty when the calculation starts.

Sometimes it may be necessary to use the same subroutine successively with a number of different parameters, as, for example, when a set of numbers, fed into the machine in coded-decimal form, are to be converted to binary scale. In this case the parameters are stored in temporary locations and transferred one by one to the appropriate subroutine by means of an iterative routine which, at the same time, counts the number of times the subroutine has been used.

Parameters which are not initially known must be inserted into the appropriate subroutines as they become available by means of 'linking' orders which form part of the main programme, and which will vary from one problem to another. Examples of this will be found in subsequent programmes.

A FURTHER CODED EXAMPLE—DIVISION

Some of the points discussed in the previous section will be illustrated here by the insertion of a division subroutine into the coded example of p. 156 in place of the automatic division order.

It is first necessary, however, to consider the programming of division in more detail. One process is to use the iteration formula:

$$x_{n+1} = x_n(2 - mx_n)$$

where $$Lt_{n\to\infty} x_n = 1/m$$

to evaluate the reciprocal of the divisor.

This method has the disadvantage that two multiplications are required for each iteration, and multiplication on *A.P.E.*(*X.*)*C.* is a comparatively slow process. For this reason an alternative method which is, in effect, the non-restoring method of division applied to the binary scale is used.

Thus, suppose that it is required to form the quotient $q = a/m$, and suppose that the remainder at any stage of the division is r. Denoting the sign digits of a, m and r by $a(s)$, $m(s)$, $r(s)$ application of the following process will give q:

Operations 1 to 31.

If $r(s) = m(s)$	Add 2^{-31} to q
	Shift left r and q
	Subtract m from r
If $r(s) \neq m(s)$	Add 0 to q
	Shift left r and q
	Add m to r
Operation 32.	Add $2^0 + 2^{-31}$ to q

The last operation effects the round off by making $q(32)$ unity; since $q(33)$ is unknown, this gives the most unbiased round off possible.

It will be seen that this division procedure produces the quotient a/m directly, whereas the iterative method requires an extra multiplication to give this result. Before using either method it is necessary to ensure that $m > a$, so that the quotient will not be outside the range of the machine. This is a distinct problem, however, and it will be discussed elsewhere.

The detailed programme can now be given. It will be assumed that the orders are stored on drum (*d*) in locations m_2 onwards; the programme is then:

Division. Forms a/m.

Add m_2 to all memory locations.

Location	Contents	Remarks
0	(23)→*cR*(1)	Iteration index placed in register
1	+(21)→*cA*(2)	*m*
2	*Cc*⩾0(4)<0(3)	
3	+(24)→*cA*(5)	0 Placed in position (20) if *m* negative
4	+(25)→*cA*(5)	1 Placed in position (20) if *m* positive
5	*A*→(20)(6)	

Location	Contents	Remarks
6	$+(22)\rightarrow cA(7)$	a
7	$+(20)\rightarrow A(8)$	Sign of a changed if m is positive
8	$Cc\geqslant 0(11)<0(9)$	
9	$l(1)(10)$	Quotient digit 1 placed in register and remainder doubled
10	$-(21)\rightarrow A(13)$	m subtracted from remainder
11	$l(1)(12)$	Quotient digit 0 placed in R and remainder doubled
12	$+(21)\rightarrow A(13)$	m added to remainder
13	$l(32)(14)$	Iteration index shifted from register to accumulator
14	$Cc\geqslant 0(15)<0(16)$	Iteration index examined
15	$l(32)(7)$	Iteration index returned to register and remainder to accumulator. Control switched to (7) to start next cycle
16	$(24)\rightarrow cR(17)$	Register cleared to avoid possible digit transfer to accumulator in (17)
17	$l(1)(18)$	Iteration index removed
18	$+(26)\rightarrow A(19)$	Sign adjusted and quotient rounded off
19	$C\rightarrow(\,.\,.)$	Control transferred to main programme
20	Temporary storage for correction digit to $A(1)$	
21	m	
22	a	
23	2^{-31}	
24	0	
25	1	
26	$1\cdot 00\ldots 001$ (i.e. $1+2^{-31}$)	

It will be seen that the quotient is available in the accumulator and that order (m_2+19) is a bridging order which enables the subroutine to be connected with the main programme. Before use, this order must be modified to link up with the rest of the programme, and it is also necessary to insert the appropriate values of a and m.

THE INCLUSION OF A SUBROUTINE IN A PROGRAMME

It is now possible to give a detailed illustration of the process of inserting a subroutine into a programme. As an example the division subroutine, given above, will be used to replace the automatic division order in the example of p. 156. The latter will, for

convenience, be referred to as the main routine. It will be assumed that the complete calculation consists of the evaluation of n terms of the form $\frac{ax^2+bx+c}{dx+e}$, and that after this is completed the machine is to stop. In general, any particular subroutine will not be stored on the same drum as the main programme, and it will therefore be assumed that the main programme is situated on drum (1) and the division subroutine on drum (d).

The following modifications are then necessary:

(i) Order (m_2+19) of the division subroutine must be modified to direct control back to the main routine.

(ii) The appropriate values of a and m must be inserted into the division subroutine.

(iii) Order (m_1+10) of the main routine must be replaced by a bridging order transferring control to the division subroutine.

(iv) Order (m_1+11) must be altered to $A\rightarrow M(l)$, since the quotient is produced in the accumulator.

(v) A 'stop' order must end the calculation at the appropriate point.

It should be noted that modification (i) need only be made once, and can therefore be outside the main iteration, whereas (ii) must be made anew each time a cycle is performed. (iii), (iv) and (v) are not subroutine modifications in the true sense, since normally they would have been incorporated in the main programme when first coded.

Finally the exact method of modifying the linking order (m_2+19) must be mentioned. In practice it is more convenient to re-insert this order completely each time the subroutine is used, rather than merely to modify the drum and memory location numbers. This does not result in any additional complication or use of memory space apart from the fact that it is necessary to store the complete order in the main routine, and not merely the drum and memory location specifications.

The complete programme is as follows:

Tabulation of $y=ax^2+bx+c/dx+e$ *using division subroutine*

Add m_1 to all memory locations within main programme.

Location	Contents	Remarks
0	$(38)\rightarrow cR(1)$	Modified form of linking order (m_2+19) in division routine
1	$R\rightarrow(m_2+19)(2)$	Order transferred to division routine

Location	Contents	Remarks
2	$(31) \rightarrow cR(3)$	d
3	$(37) \times R \rightarrow cA(4)$	dx
4	$+(32) \rightarrow A(5)$	$dx+e$
5	$r(32)(6)$	$(dx+e)$ transferred to register
6	$R \rightarrow (m_2+21)(7)$	Divisor transferred to division routine
7	$(28) \rightarrow cR(8)$	a
8	$(37) \times R \rightarrow cA(9)$	ax
9	$+(29) \rightarrow A(10)$	$ax+b$
10	$r(32)(11)$	$(ax+b)$ transferred to register
11	$(37) \times R \rightarrow cA(12)$	ax^2+bx
12	$+(30) \rightarrow A(13)$	ax^2+bx+c
13	$r(32)(14)$	(ax^2+bx+c) transferred to register
14	$R \rightarrow (m_2+22)(15)$	Dividend to division routine
15	$C \rightarrow (m_2)$	Control to division routine to give $ax^2+bx+c/dx+e$
16	$A \rightarrow (l)(17)$	Term stored in memory
17	$+(36) \rightarrow cA(18)$	I
18	$-(35) \rightarrow A(19)$	$I-2^{-31}$
19	$Cc \geqslant 0(20) < 0(27)$	Sign of I tested
20	$A \rightarrow (36)(21)$	New index substituted
21	$+(37) \rightarrow cA(22)$	x
22	$+(33) \rightarrow A(23)$	$x+\triangle$
23	$A \rightarrow (37)(24)$	New value of x substituted
24	$+(16) \rightarrow cA(25)$	y storage order
25	$+(34) \rightarrow A(26)$	2^{-8} added to x location of storage order
26	$A \rightarrow (16)(2)$	Storage order replaced, and control sent to start of iterative cycle
27	Stop	
28	a	
29	b	
30	c	
31	d	
32	e	
33	$\triangle$	
34	2^{-8}	
35	2^{-31}	
36	I	
37	x	

Location	Contents	Remarks
38	$C \rightarrow (m_1 + 16)$	Transfer link for end of division subroutine
$l - (l+n)$		Table of values of y as computed

THE CONTROL OF THE MAGNITUDE OF NUMBERS

It has already been mentioned that the programmer for a 'fixed point' machine must consider the magnitude of the numbers which will be generated in the course of a calculation, and make sure that these are not allowed to increase or diminish beyond the range of the machine. Sometimes it will be possible to do this beforehand by multiplying the parameters of the problem by suitable scale factors, but often the magnitude of all the numbers produced in the course of a calculation cannot be foreseen. Again, if the calculation is not too involved, comparatively simple programme checks may suffice. Thus, in the example of p. 156, suppose that it is not known beforehand that $(dx+e) > (ax^2+bx+c)$ for all values of x, and that both factors may be either positive or negative.

The following programme check compares the sizes of numerator and denominator (written n and d for convenience); if $n > d$, d is multiplied by two and a fresh comparison made, and this process is repeated until $n < d$. At the same time, a record is made for future reference of the number of times d is increased. The programme is based upon the fact that if the quantities $(n+d)$, $(n-d)$ are formed, they will have the same sign if $n > d$, but different signs if $n < d$.

The programme is as follows:

Control of magnitude of numbers in evaluating n/d

Add m_3 to all memory locations.

Location	Contents	Remarks
0	$(23) \rightarrow cR(1)$	Clears register to zero
1	$R \rightarrow (27)(2)$	Shift counter cleared to zero
2	$+(26) \rightarrow cA(3)$	$+d$
3	$+(25) \rightarrow A(4)$	$n+d$
4	$Cc \geqslant 0(5) < 0(7)$	
5	$+(25) \rightarrow cA(6)$	n
6	$-(26) \rightarrow A(9)$	$n-d$
7	$-(25) \rightarrow cA(8)$	$-n$
8	$+(26) \rightarrow A(9)$	$+d-n$ (to unify conditional transfer (9))
9	$Cc \geqslant 0(10) < 0(22)$	Programme complete if $n < d$

Location	Contents	Remarks
10	$+(27)\rightarrow cA(11)$	Shift counting index increased by 1 and examined. Stop (21) if more than 31 shifts have occurred
11	$+(24)\rightarrow A(12)$	
12	$Cc\geqslant 0(13)<0(21)$	
13	$A\rightarrow(27)(14)$	New index inserted in store
14	$+(26)\rightarrow cA(15)$	d
15	$Cc\geqslant 0(16)<0(18)$	
16	$l(1)(17)$	$2d$
17	$Cc\geqslant 0(20)<0(21)$	
18	$l(1)(19)$	$2d$
19	$Cc\geqslant 0(21)<0(20)$	
20	$A\rightarrow(26)(3)$	
21	Stop	Indicates an unforeseen event, either $d=0$ or a digit must be lost from n in order to obtain $d>n$
22	$C\rightarrow(\ )$	Transfer to main routine
23	0	
24	2^{-5}	
25	n	
26	d	
27	Shift count	

Notice that $(d-n)$ has been formed at operation (m_3+8) instead of $(n-d)$. This is to unify the discrimination (m_3+9) when $(n+d)$ has a negative sign. Several possible contingencies have also been anticipated, it is possible that both n and d consist of 31 binary digits having digits 1 and 31 unity. If $n>d$, in these circumstances, the only way to produce $n^1<d$ is to right shift n one place thus losing a significant digit, since the programmer should be made aware of such loss of accuracy the programme is halted by (m_3+19). Another potential source of error occurs when $d=0$, the division of n by d being then meaningless, this is detected by (m_3+12) and the programme again arrested.

The limitations of such simple checks are immediately obvious, for if the result of the division (n/d) is to be used in subsequent calculations, it will be necessary to modify the programme to take account of the altered value of n/d. This illustrates a general principle, namely that, if at any point in a calculation account is taken of the size of the numbers, then it must also be taken all through the calculation. Thus, it is possible to devise subroutines for each arithmetic process which operate on numbers expressed in the form $x\times 2^p$, but in general, these 'floating point' routines as they may be called, must either be used all through the calculation or

not at all. Such subroutines are rather lengthy, and will not be further discussed for that reason; details have been given elsewhere by Wilkes, Wheeler and Gill. The most serious objection to their use is that the effective operating time of the machine is thereby greatly reduced; thus Wilkes estimates that the time of multiplication is increased from 2·5 msec to 50 msec. Moreover, coding is complicated since each arithmetic operation involves reference to a subroutine. Thus, although floating point arithmetic is available, it should be regarded as a last resort rather than as an easy alternative to careful planning.

It may be remarked here, that the most serious problem is that of detecting when numbers become too large for the machine, since quantities which diminish below the range can usually (although not always) be regarded as small enough to be neglected anyway. Unfortunately, there is no direct means of accomplishing this in a machine using complementary form for negative numbers, since a positive number which has exceeded the range is indistinguishable from a negative number. It is possible, by printing results at selected points during the calculation, to keep a check on the size of the number produced. It will then be obvious if a function has exceeded the range of the machine for it will suddenly change from, say, positive increasing to a small negative value. This method suffers the same disadvantage as the previous one, for printing is a slow operation and the overall speed of the machine will be considerably reduced. Moreover, the trouble is merely detected and in order to cure it the calculation must be repeated after suitable adjustments. It may, however, be useful when a number of identical calculations are required with different parameters, and where it is known that the numbers generated in each will be similar in size. Preliminary adjustment in one case on the lines indicated above will then enable the remainder to be run through at normal speed.

In fact, the most satisfactory solution from the programmer's point of view is a machine built to work with a floating point. Unfortunately the engineering difficulties encountered in this type of machine are great, and a considerable increase in the size and complexity of the 'hardware' seems inevitable. Meanwhile, the problem of the control of numbers remains one of the most unsatisfactory features of fixed point automatic computers.

THE EXTRACTION OF THE SQUARE ROOT

An important arithmetic operation which is not generally provided as a machine function is the extraction of the square root.

A number of different processes exist, most of which are readily coded for a digital computer. Among these are:

(1) The standard process taught in schools.
(2) Taylor series expansion.
(3) Simple iteration.
(4) Two stage iteration.
(5) Comparison and improvement.
(6) Series approximation.
(7) Interpolation from tabular entries.

It is not profitable to show detailed codes for each of the above possibilities and only the mathematical bases will be given, together with a detailed code of two variants which have been used on the *A.P.E.*(*X.*)*C.*

The binary form of the school arithmetic process is identical, in principle, with that used for decimals. The number whose root is required is divided into periods of two digits and a trial root is derived for the first of these. In scale of two this trial is fortunately 0 or 1. After subtraction of the square of the trial and examination of the sign of the remainder, the trial is multiplied by two and used for the first digit pair of a second stage of the process—just as in ordinary arithmetic. The method will be clear from the following example:

$$49/64 = (7/8)^2 = \cdot 110001$$

	$\cdot 1\ \dot{1}\ 0\ \dot{0}\ 0\ \dot{1}$ $(\cdot 111$
1	1
	1 0 0 0
1 0 1	1 0 1
	0 0 1 1 0 1
1 1 0 1	1 1 0 1
	. . . . .

This process is efficient, from the machine point of view, since successive approximations are obtained without multiplication (except, of course, for the doubling, which is trivial in binary scale).

The Taylor series approach depends on the expansion:

$$(1-x)^{\frac{1}{2}} = 1 - \tfrac{1}{2}x - \tfrac{1}{8}x^2 - \tfrac{1}{16}x^3 - \tfrac{5}{128}x^4 \ \ldots\ . \quad (x<1)$$

it is simple to code but has the disadvantage that the number of

terms required to produce a given accuracy depends upon the value of x.

Possibly the simplest iterative process depends upon the formula:

$$x_{n+1} = \left(x_n + \frac{b}{x_n}\right) \Big/ 2$$

where $$Lt_{n\to\infty} x_n = \sqrt{b}$$

which defines an iterative process for the calculation of $\sqrt{b}$.

The programme for this calculation is one of the more frequently used subroutines, particularly in x-ray crystallography and ballistics. It also serves as an illustration of the way in which calculations must sometimes be rearranged in order to avoid the introduction of quantities greater than unity.

First the magnitude of the term b/x_n must be examined. If ε is the error in the nth approximation, it can be shown that the error in the $(n+1)$th is $\varepsilon^2/2\sqrt{b}$. Since $b<1$ it follows that:

$$x_{n+1} > \sqrt{b} > b$$

and therefore $b/x_{n+1}<1$ and is within the range of the machine. The first approximation will be taken as $(b+1)/2$ and this is no exception to the rule.

The term $(x_n + b/x_n)$ may, however, be >1 but this difficulty can be avoided by calculating:

$$\tfrac{1}{2}\left(\frac{b}{x_n} - x_n\right) + x_n$$

This form is chosen rather than

$$\frac{x_n}{2} + \frac{b}{2x_n}$$

since the rounding off errors are less. Moreover, the value of $\left(\frac{b}{x_n} - x_n\right)\Big/2$ can be used to decide whether the approximation has reached the limits of accuracy of the machine since it is equal to $x_{n+1} - x_n$. When this point is reached $(x_{n+1} - x_n)$ will change from a negative quantity to zero. In fact, a slightly modified procedure is followed, for if the criterion outlined above were used, the result would always be calculated to the full accuracy of the machine's capacity. This is not always necessary and might result in wasted time. The calculation can, however, be stopped when any predetermined accuracy is reached by adding 2^{-k} to $(x_{n+1} - x_n)$ after

the calculation of each term, where k defines the accuracy required, and discriminating upon the result.

Code for Square Root iteration. $(x_{n+1} = \frac{1}{2}(b/x_n - x_n) + x_n)$

Add m_4 to all memory locations.

Location	Contents	Remarks
0	$(20) \rightarrow cR(1)$	
1	$R \rightarrow (m_2 + 19)(2)$	Last order of division subroutine modified
2	$+(25) \rightarrow cA(3)$	b
3	$r(1)(4)$	$b/2$
4	$+(21) \rightarrow A(5)$	$(b+1)/2 = x_1$
5	$A \rightarrow (22)(6)$	x_1 initially, x_n at later stage
6	$r(32)(7)$	x_n transferred to register
7	$R \rightarrow (m_2 + 21)(8)$	x_n to division subroutine
8	$(25) \rightarrow cR(9)$	b
9	$R \rightarrow (m_2 + 22)(10)$	b to division subroutine
10	$C \rightarrow (m_2)$	Division subroutine
11	$-(22) \rightarrow A(12)$	$b/x_n - x_n$
12	$r(1)(13)$	$\frac{1}{2}(b/x_n - x_n) = x_{n+1} - x_n$
13	$A \rightarrow (23)(14)$	
14	$+(24) \rightarrow A(15)$	$2^{-k} + (x_{n+1} - x_n)$ negative until precision 2^{-k} is reached
15	$Cc \geqslant 0(18) < 0(16)$	
16	$+(23) \rightarrow cA(17)$	$x_{n+1} - x_n$ to accumulator
17	$+(22) \rightarrow A(5)$	x_{n+1} in accumulator
18	$+(22) \rightarrow cA(19)$	x_n—the required square root—in the accumulator
19	$C \rightarrow (\ )$	
20	$C \rightarrow (m_4 + 11)$	
21	$\frac{1}{2}$	
22	x_n	
23		Temporary storage
24	2^{-k}	Defines accuracy of result
25	b	

Parameters of the programme are b and k and the final result appears in the accumulator.

The number of iterations required for any given accuracy depends on the size of b, increasing as b decreases. Thus, suppose that the result is required correct to 6 decimal digits. In the worst possible case, when $b = 0$, 31 iterations are needed; on the other hand, when,

b is large ($=0{\cdot}9$, say) 3 iterations suffice. It follows that if a large number of square roots are required much time will be saved if the numbers are adjusted, either before insertion into the computer, or by the programme itself, to make them as large as possible.

It will be seen that the above method has the disadvantage that a division is required at each stage and is not well suited to a machine lacking an automatic divider. A second variant is:

$$x_{n+1} = x_n[3b - x_n^2]/2b$$

which also converges to $\sqrt{b}$ but requires only a single division to calculate $1/2b$. The disadvantage lies, however, in the plurality of multiplications required to calculate the numerator.

By the use of a multiple iteration the root can be found without division. Thus:

$$x_{n+1} = x_n - \tfrac{1}{2}x_n y_n$$
$$y_{n+1} = y_n^2[y_n - 3]/4$$

with $x_0 = b, y_0 = b - 1$

converges to:

$$\underset{n\to\infty}{Lt}\ x_n = \sqrt{b}$$

$$\underset{n\to\infty}{Lt}\ y_n = 0$$

Again, several multiplications are required at each stage.

A process which leads to a particularly simple programme and is easily extended to higher order roots depends upon the building up of the root one digit at a time. To start the cycle $0{\cdot}1$ is squared and the number b whose root is required is subtracted from it. If the resulting number is positive $0{\cdot}1$ is too large and $0{\cdot}01$ is subtracted from it to give the next trial approximation; if the difference is negative, however, $0{\cdot}01$ is added to the first approximation to continue the process. Although this method is not the sort which would be contemplated in hand calculation, it is very suitable for a digital computer provided with a high speed multiplier and leads to an elegant programme. It should be noticed that the first approximation must always be $\frac{1}{2}$, and that the process depends upon the fact, used in the non-restoring multiplication and division procedures, that $2^{-n} - 2^{-n-1} = 2^{-n-1}$(i.e. $2 - 1 = 1$!).

An ingenious root extraction process, used in the *E.N.I.A.C.*, depends upon the fact that the sum of the first n odd numbers is equal to n^2, i.e.

$$1 + 3 + 5 + 7 \ \ldots\ . + (2n - 1) = n^2$$

This is easily coded, involves no multiplications, but is very slow for large values of n.

The method of storing a table of square roots and interpolating from it to obtain the given root is obviously applicable, but in view of the complexity of the interpolation process discussed in another part of this section it is not often used. A more satisfactory use of a stored table of roots is to obtain a good first approximation for one of the iterative techniques previously discussed. These iterations are such that the error at any stage is the square of that at the previous stage, so that rapid convergence depends upon a good initial value.

As an example of a typical square root programme the comparison/improvement process is given below.

Square root—Successive Trial and Comparison

Add m_4 to all memory locations.

Location	Contents	Remarks
0	$+(19)\rightarrow cA(1)$	$2^{-1}=0{\cdot}100\ldots$
1	$A\rightarrow(23)$ (2)	Probe and approximation restored after previous use of routine
2	$+(20)\rightarrow cA(3)$	0 (zero)
3	$A\rightarrow(21)$ (4)	Initial value of root replaces residue from last use of routine
4	$(21)\rightarrow R(5)$	Approximation to root, x_n, say
5	$(21)\times R\rightarrow cA(6)$	x_n^2
6	$-(22)\rightarrow A(7)$	x_n^2-b
7	$Cc\geqslant 0(8)<0(9)$	
8	$-(23)\rightarrow cA(10)$	x_n decreased to form x_{n+1}
9	$+(23)\rightarrow cA(10)$	x_n increased to form x_{n+1}
10	$+(21)\rightarrow A(11)$	
11	$A\rightarrow(21)$ (12)	x_n replaced by x_{n+1}
12	$+(23)\rightarrow cA(13)$	
13	$r(1)$ (14)	Probe digit shifted one place to right
14	$A\rightarrow(23)$ (15)	New probe substituted
15	$l(32)$ (16)	New probe shifted and used as iteration index
16	$Cc\geqslant 0(4)<0(17)$	If probe shifted out of accumulator process completed
17	$(21)\rightarrow cA(18)$	Square root placed in accumulator

Location	Contents	Remarks
18	$C\rightarrow(\ )$	Control transferred to main programme
19	0·100 . . .	
20	0·000 . . .	
21	x	
22	b	
23	i	

A point of interest lies in the use of the probe digit, not only to form the next approximation, but also to act as an iteration index. This is a consequence of the fact that, in right shift, digits flow from the least significant accumulator digit to the most significant register digit. Thus, when the precision of approximation exceeds the accumulator capacity, the probe digit appears in $R(1)$, is transferred to $A(1)$ by the $l(32)$ operation (15) and can be used, with the conditional transfer (16) to terminate the process. A simple extension of the programme enables n multiplications to be performed instead of the single one at (5), and in this case the result of the iteration is the formation of $\sqrt[n]{b}$.

THE CONVERSION OF NUMBERS TO AND FROM BINARY SCALE

The next two examples are most important from the practical point of view, since they are concerned with the conversion of numbers to and from the binary scale, and will normally be used in every calculation.

DECIMAL–BINARY CONVERSION

The normal arithmetic process for decimal–binary conversion of a number less than 1 is as follows:

The number is successively multiplied by 2, and 1 or 0 is recorded in the corresponding binary number according as there is or is not a carry past the decimal point. Unfortunately this method is not very suitable for a calculating machine since decimal digits are not inserted as such, but are represented in coded binary form. The conversion process outlined above becomes very complicated when applied to a number expressed in coded binary form, since carries between the digits must be adjusted at each multiplication.

It is more convenient to adopt the less direct method now to be described. This is best illustrated by an example; thus suppose

the number 0·431 is to be converted to the binary scale. This can be written:

$$10^{-3}((4.10+3)10+1)$$

The process can be analysed as follows:

Multiply the first digit of the number by 10 (1010 in binary scale), add in the next digit and multiply by 10, add in the next digit, *etc.* This process is repeated until all the digits have been used. Finally the result is multiplied by 10^{-n}, where n is the number of digits. In order to keep the numbers within the range of the machine, the decimal digits must be added into the least significant end of the accumulator. At the end of the calculation it will therefore be necessary to divide the result by a factor $10^n/2^{31}$. The number of digits, n, may vary from one calculation to another, the maximum value for *A.P.E.*(*X.*)*C.* being 8, and the programme is therefore constructed so that n may assume any value between 0 and 8.

One more point must be mentioned, concerning the sign of the number. Negative decimal numbers will normally be inserted as an absolute value together with a sign digit unity. The conversion process can therefore be applied directly to them, but after the conversion is finished the complement must be taken and the sign digit made unity in accordance with the convention adopted for negative numbers within the machine.

Code for Decimal–Binary conversion

Add m_5 to all memory locations.

Location	Contents	Remarks
0	$+(35)\rightarrow cR(1)$	Clears register
1	$R\rightarrow(39)(2)$	Storage location for partially converted number cleared
2	$+(36)\rightarrow cA(3)$	Decimal number
3	$Cc\geqslant 0(5)<0(4)$	
4	$+(37)\rightarrow A(5)$	Sign of D deleted if required
5	$l(1)(6)$	
6	$l(4)(7)$	First decimal digit of D sent to register
7	$A\rightarrow(38)(8)$	Unconverted part of D stored
8	$l(32)(9)$	Decimal digit of D (or unconverted part) to accumulator
9	$+(35)\rightarrow cR(10)$	Register cleared
10	$+(39)\rightarrow A(11)$	Part already converted added in
11	$A\rightarrow(39)(12)$	

Location	Contents	Remarks
12	$l(2)(13)$	
13	$+(39)\rightarrow A(14)$	
14	$l(1)(15)$	
15	$A\rightarrow(39)(16)$	
16	$+(40)\rightarrow cA(17)$	Iteration index
17	$-(41)\rightarrow A(18)$	Unity subtracted
18	$A\rightarrow(40)(19)$	New index stored
19	$Cc\geqslant 0(20)<0(22)$	
20	$(35)\rightarrow cR(21)$	Register cleared
21	$+(38)\rightarrow cA(6)$	Unconverted part of decimal number. Control directed to start of next cycle
22	$(39)\rightarrow cR(23)$	Converted number
23	$R\rightarrow(m_2+22)(24)$	Substitution in division subroutine
24	$(42)\rightarrow cR(25)$	Divisor.
25	$R\rightarrow(m_2+21)(26)$	Substitution in division subroutine
26	$(43)\rightarrow cR(27)$	Output transfer order
27	$R\rightarrow(m_2+19)(28)$	Substitution in division subroutine
28	$C\rightarrow(m_2)$	Control to start of division subroutine
29	$A\rightarrow(38)(30)$	Binary digits stored
30	$+(36)\rightarrow cA(31)$	D
31	$Cc\geqslant 0(32)<0(33)$	
32	$+(38)\rightarrow cA(34)$	
33	$-(38)\rightarrow cA(34)$	
34	$C\rightarrow(\ )$	
35	0	
36	D	Decimal number to be converted
37	1	
38	—	Temporary storage for unconverted part of D
39	—	Partially converted number
40	I	Iteration index $(n-1)\times 2^{-31}$ initially n = number of decimal digits
41	2^{-31}	
42	$10^n\times 2^{-31}$	
43	$C\rightarrow(m_5+29)$	Output instruction for division subroutine

Parameters which must be inserted are D, the iteration index $(n-1)\times 2^{-31}$ and the scale factor $10^n\times 2^{-31}$. The result appears in the accumulator.

Binary–Decimal conversion

Binary–decimal conversion is slightly simpler than the reverse process as the straightforward arithmetic procedure can be used. Moreover, as no division occurs, other subroutines are not involved.

The binary number is successively multiplied by 10×2^{-4} ($\cdot 1010$ in binary scale). The corresponding decimal digits are then given, in coded binary form, by the carry past the binary point when this product is multiplied by 2^4. The absolute value of the binary number must be used during conversion, and a negative sign attached at the end if appropriate.

Programme for Binary to decimal conversion

Add m_6 to all memory locations.

Location	Contents	Remarks
0	$+(23) \rightarrow cA(1)$	Binary number. B
1	$Cc \geqslant 0(2) < 0(3)$	
2	$(20) \rightarrow cR(5)$	If B positive 0 sent to 32nd stage of register
3	$-(23) \rightarrow cA(4)$	Absolute value of B obtained
4	$(21) \rightarrow cR(5)$	If B negative 1 sent to 32nd stage of register
5	$R \rightarrow (22)(6)$	Storage for decimal digits
6	$A \rightarrow (23)(7)$	$\lvert B \rvert$
7	$r(2)(8)$	$\lvert B \rvert/4$
8	$+(23) \rightarrow A(9)$	$\lvert B \rvert + \lvert B \rvert/4 = (\frac{1}{2})^3 \times 10 \lvert B \rvert$
9	$(22) \rightarrow cR(10)$	
10	$l(4)(11)$	$10 \lvert B \rvert$ shifted into R
11	$R \rightarrow (22)(12)$	Partial decimal stored
12	$l(63)(13)$	Equivalent to one right shift of accumulator
13	$A \rightarrow (23)(14)$	Binary remainder stored
14	$+(24) \rightarrow cA(15)$	
15	$-(21) \rightarrow A(16)$	
16	$Cc \geqslant 0(17) < 0(19)$	
17	$A \rightarrow (24)(18)$	
18	$+(23) \rightarrow cA(7)$	
19	$C \rightarrow (\ )$	
20	0	
21	2^{-31}	

Location	Contents	Remarks
22	Decimal number as generated	
23	Binary number originally and then unconverted residue	
24	Counting index for number of decimal digits	

Parameters of the programme are the binary number B and the iteration index $(n-1)\times 2^{-31}$. The result appears in the accumulator.

16

PROGRAMME DESIGN

THE examples of programmes examined in the previous chapters have been relatively short and of a simple character. To give the reader a more complete picture of the considerations which obtain in the programming of a real problem, it is proposed to deal next with several more complicated situations. This will show how the procedure must be varied in order to make the best possible use of the machine in any given calculation, and also that computational means which are appropriate in one use of a certain function may not be the best when exactly the same function is used in another context.

THE USE OF TABLES AND INTERPOLATION

Mathematical tables are an important element in computing, and it is essential that methods should be available for using them in conjunction with the calculating machine, and for interpolating from them when necessary.

A large, permanent memory is available with *A.P.E.*(*X.*)*C.*, and the frequently used functions such as sine, tan and exponential are stored there. Any function which is not available in this way can either be inserted when needed, or, if only one or two values are required, these may be calculated directly by means of a Taylor or some similar series.

THE STORAGE OF TABLES

It is appropriate, at this point, to describe the method of storing tables in the memory of *A.P.E.*(*X.*)*C.* A tabulated function consists of two sets of figures, the function itself and the argument. It is possible to use two memory locations for each entry in a table, storing values of function and argument alternately in consecutive positions, but this scheme is uneconomical from the viewpoint of memory space. Alternatively, one memory location can be made to serve for both function and argument, but it is then necessary to arrange for the separation of the two numbers before the function

can be used. Moreover, the maximum possible accuracy of the table is reduced.

An alternative approach is suggested when it is remembered that functions are almost invariably tabulated at equal intervals of the argument. Since successive memory locations are identified by numbers differing by equal intervals 2^{-n} it is possible to arrange that the first m digits of the argument (usually plus a constant) are equal to the number giving the memory location in which the corresponding function is stored. If the argument corresponds exactly to an entry in the table this can then be located by substituting the argument in the memory location digits of an order which will extract the appropriate value of the function from the memory. If there is no entry corresponding exactly to the argument it can be used in the same way to extract the nearest existing entry in the table.

Thus, to give an example (expressed in decimal scale for simplicity), the function e^x for $x=0{\cdot}125$ will be stored in memory location $125+c$ where c is constant for the whole table. If the value of $e^{0{\cdot}1252}$ is required it will be found by interpolation using the entries around $M(125+c)$ as a basis.

This method of storage will sometimes necessitate adjustment of the arguments, but the machine can be programmed to effect these. Another consequence is that functions may have to be tabulated at rather unconventional intervals. For example, it is convenient to tabulate the trigonometric functions at intervals of $\pi/2^n$, but since most tables will be computed by the machine itself, this is not a serious disadvantage.

A PROGRAMME FOR INTERPOLATION

It follows from the previous discussion that a general interpolation routine will be a necessary member of the collection of subroutines. Stirling's formula has been chosen as an example as this uses central differences lying as closely as possible to the required value. It will, of course, be necessary to use other methods under some circumstances; for example, when interpolating near the end of the table, Newton's formulae are more appropriate. The coding for these cases will be very similar, however, and will not be given here.

The accuracy of an interpolation depends on the number of terms used, and for any given accuracy this will vary from one function to another. The ideal interpolation programme would be quite general, and capable of producing a result to any pre-assigned degree of accuracy, irrespective of the number of terms needed for

any particular function. Unfortunately the coding of such a universal method is very lengthy since, in order to make such a process iterative, it is necessary to work from the difference tables of the function rather than to calculate each term of the interpolation individually. As, in general, there is no means of deciding beforehand how many terms will be required, the formation of these difference tables is an unwieldy process unless the whole of each table is calculated as needed, and this may occupy much time and memory space.

It was therefore decided to confine the programme to five terms. This will probably be adequate for most purposes; for example, interpolating from a table of $\sin(x)$ at intervals of $(\pi/32)$, and taking five terms, the result is obtained correct to 6 decimal places. The programme is so arranged that the calculation is stopped when a previously specified degree of accuracy is reached with less than five terms, thus avoiding time wastage.

Suppose, now, that a function y is tabulated at intervals ω of the argument x, and that it is required to interpolate for $x = x_0 + n\omega$, where x_0 is the nearest smaller tabulated value to x. Successive positions above and below in the table are x_{-1}, x_{+1}, x_{-2}, *etc.*

Stirling's formula to five terms is:

$$y = y_0 + \frac{n}{2}[y_{+1} - y_{-1}] + \frac{n^2}{2!}[y_{+1} - 2y_0 + y_{-1}]$$

$$+ \frac{n(n^2-1)}{2.3!}[y_{+2} - 2y_{+1} + 2y_{-1} - y_{-2}]$$

$$+ \frac{n^2(n^2-1)}{4!}[y_{+2} - 4y_{+1} + 6y_0 - 4y_{-1} + y_{-2}]$$

Code for Interpolation routine

Add m_7 to all memory locations.

Location	Contents	Remarks
0	$+(93) \rightarrow cA(1)$	x (assumed positive)
1	$r(22)(2)$	Proportional part shifted into register
2	$+(109) \rightarrow cA(3)$	Accumulator cleared
3	$r(1)(4)$	n = proportional part/ω in register
4	$R \rightarrow (94)(5)$	n stored for future use
5	$-(94) \rightarrow cA(6)$	If $n > 0$ this gives a negative number in accumulator
6	$Cc \geqslant 0(7) < 0(13)$	Tests if x is an exact entry in the table
7	$+(93) \rightarrow cA(8)$	x

Location	Contents	Remarks
8	$l(1)(9)$	The first nine digits of x are now correctly placed for substitution in an order
9	$+(95)\rightarrow A(10)$	Constant c giving location of first tabular entry added
10	$A(1\text{–}9)\rightarrow(11)(11)$	Order (11) modified
11	$+(y_0)\rightarrow cA(12)$	The required value of y is extracted from the table
12	$A\rightarrow(96)(92)$	Tabular value stored in output location
13	$+(93)\rightarrow cA(14)$	x (not an exact entry)
14	$l(1)(15)$	Adjusted, as before, for substitution
15	$+(95)\rightarrow A(16)$	c added
16	$A(1\text{–}9)\rightarrow(32)(17)$	Order selecting y_0 modified
17	$+(97)\rightarrow A(18)$	2^{-8}
18	$A(1\text{–}9)\rightarrow(35)(19)$	Order selecting y_{+1} modified
19	$A(1\text{–}9)\rightarrow(52)(20)$	Order selecting y_{+1} modified
20	$+(97)\rightarrow A(21)$	2^{-8}
21	$A(1\text{–}9)\rightarrow(60)(22)$	Order selecting y_{+2} modified
22	$A(1\text{–}9)\rightarrow(72)(23)$	Order selecting y_{+2} modified
23	$-(98)\rightarrow A(24)$	-3×2^{-8}
24	$A(1\text{–}9)\rightarrow(34)(25)$	Order selecting y_{-1} modified
25	$-(97)\rightarrow A(26)$	-2^{-8}
26	$A(1\text{–}9)\rightarrow(59)(27)$	Order selecting y_{-2} modified
27	$(94)\rightarrow cR(28)$	n
28	$(94)\times R\rightarrow cA(29)$	n^2
29	$A\rightarrow(99)(30)$	n^2 stored
30	$-(100)\rightarrow A(31)$	(n^2-1)
31	$A\rightarrow(101)(32)$	(n^2-1) stored
32	$+(y_0)\rightarrow cA(33)$	
33	$A\rightarrow(96)(34)$	y_0 stored
34	$-(y_{-1})\rightarrow cA(35)$	$-y_{-1}$
35	$+(y_{+1})\rightarrow A(36)$	$y_{+1}-y_{-1}$
36	$r(1)(37)$	$(y_{+1}-y_{-1})/2$
37	$A\rightarrow(102)(38)$	
38	$r(32)(39)$	
39	$(94)\times R\rightarrow cA(40)$	$\frac{n}{2}(y_{+1}-y_{-1})$
40	$A\rightarrow(103)(41)$	
41	$+(104)\rightarrow A(42)$	Previous terms of interpolation added in. (104) initially clear
42	$A\rightarrow(104)(43)$	New total inserted in (104)

Location	Contents	Remarks
43	$+(103)\rightarrow cA(44)$	
44	$Cc\geqslant 0(46)<0(45)$	
45	$-(103)\rightarrow cA(46)$	Positive absolute value of new term
46	$-(105)\rightarrow A(47)$	-2^{-n}, tests accuracy reached
47	$*Cc\geqslant 0(49)<0(48)$	This order is changed as the programme proceeds
48	$+(104)\rightarrow cA(86)$	Sufficient accuracy has been attained, conclusion initiated
49	$+(110)\rightarrow cA(50)$	
50	$A\rightarrow(47)(51)$	Discriminating order (47) modified for next cycle
51	$-(102)\rightarrow cA(52)$	$(y_{-1}-y_{+1})/2$
52	$+(y_{+1})\rightarrow A(53)$	$(y_{-1}+y_{+1})/2$
53	$A\rightarrow(106)(54)$	
54	$-(96)\rightarrow A(55)$	$(y_{-1}-2y_0+y_{+1})/2$
55	$r(32)(56)$	
56	$(99)\times R\rightarrow cA(41)$	$\frac{n^2}{2}(y_{+1}-2y_0+y_{-1})$, third term in series
57	$+(111)\rightarrow cA(58)$	
58	$A\rightarrow(47)(59)$	Discriminating order (47) again modified for next cycle
59	$-(y_{-2})\rightarrow cA(60)$	$-y_{-2}$
60	$+(y_{+2})\rightarrow A(61)$	$(y_{+2}-y_{-2})$
61	$r(1)(62)$	$(y_{+2}-y_{-2})/2$
62	$A\rightarrow(107)(63)$	
63	$r(1)(64)$	$(y_{+2}-y_{-2})/4$
64	$-(102)\rightarrow A(65)$	$(y_{+2}-2y_{+1}+2y_{-1}-y_{-2})/4$
65	$r(32)(66)$	
66	$(108)\times R\rightarrow cA(67)$	$(y_{+2}-2y_{+1}+2y_{-1}-y_{-2})/12$
67	$r(32)(68)$	
68	$(101)\times R\rightarrow cA(69)$	$\frac{(n^2-1)}{12}[y_{+2}-2y_{+1}+2y_{-1}-y_{-2}]$ (*N.B.* $12=2.3!$)
69	$r(32)(70)$	
70	$(94)\times R\rightarrow cA(41)$	$\frac{n(n^2-1)}{2.3!}[y_{+2}-2y_{+1}+2y_{-1}-y_{-2}]$, fourth term in the series
71	$-(107)\rightarrow cA(72)$	$(y_{-2}-y_{+2})/2$
72	$+(y_{+2})\rightarrow A(73)$	$(y_{+2}+y_{-2})/2$
73	$r(2)(74)$	$(y_{+2}+y_{-2})/8$
74	$-(106)\rightarrow A(75)$	$(y_{+2}-4y_{+1}-4y_{-1}+y_{-2})/8$

Location	Contents	Remarks
75	$r(32)(76)$	
76	$(108) \times R \rightarrow cA(77)$	$(y_{+2} - 4y_{+1} - 4y_{-1} + y_{-2})/24$ ($N.B.$ $24 = 4!$)
77	$A \rightarrow (103)(78)$	
78	$+(96) \rightarrow cA(79)$	y_0
79	$r(2)(80)$	$y_0/4$
80	$+(103) \rightarrow A(81)$	$(y_{+2} - 4y_{+1} + 6y_0 - 4y_{-1} + y_{-2})/4!$
81	$r(32)(82)$	
82	$(101) \times R \rightarrow cA(83)$	$(n^2 - 1)[y_{+2} - 4y_{+1} + 6y_0 - 4y_{-1} + y_{-2}]/4!$
83	$r(32)(84)$	
84	$(99) \times R \rightarrow cA(85)$	$\frac{n^2(n^2-1)}{4!}[y_{+2} - 4y_{+1} + 6y_0 - 4y_{-1} + y_{-2}]$ Fifth term in the series
85	$+(104) \rightarrow A(86)$	Previous terms added in
86	$+(96) \rightarrow A(87)$	y_0
87	$A \rightarrow (96)(88)$	The result recorded
88	$+(112) \rightarrow cR(89)$	
89	$R \rightarrow (47)(90)$	Order (47) restored to its original state ready for next use of subroutine
90	$+(109) \rightarrow cR(91)$	0
91	$R \rightarrow (104)(92)$	(104) is cleared
92	$C \rightarrow (\ )$	
93	x	
94	n	
95	c	Start of table
96	y_0	Initially. Finally this contains the result
97	2^{-8}	
98	3×2^{-8}	
99	n^2	
100	1	
101	$n^2 - 1$	
102	$(y_{+1} - y_{-1})/2$	
103	—	Temporary storage
104	—	Initially zero, accumulates successive terms
105	2^{-n}	n varies according to the accuracy required
106	$(y_{+1} + y_{-1})/2$	
107	$(y_{+2} - y_{-2})/2$	
108	1/3	
109	0	

Location	Contents	Remarks
110	$Cc \geqslant 0(57) < 0(48)$	
111	$Cc \geqslant 0(71) < 0(48)$	
112	$Cc \geqslant 0(49) < 0(48)$	

The three quantities x, c and 2^{-n} must be inserted in this subroutine before use; the result of the interpolation is available in the accumulator. If the interpolation is to be used several times successively it is important to make sure that the previous result is extracted and stored by the main programme.

INTERPOLATION FOR SINE OR COSINE

The preceding example, although of general application, is somewhat lengthy, and in practice it is worth while preparing shorter though more specialized routines for interpolation from functions in constant use. Probably those most frequently required are the sine and cosine and the process of interpolating from a table of sines will now be considered. This has been chosen in preference to the cosine since, as will be seen shortly, the process of deciding the sign of the function is made easier. The routine can, of course, also be used for the calculation of a cosine since $\cos(\theta) = \sin(\pi/2 - \theta)$.

Suppose, then, that it is required to calculate $\sin(\theta)$, where θ may have any value, positive or negative, and that a table of sines at some interval $\pi/2^n$ is available. It is first necessary to consider the exact form of the argument, θ. This will, in general, have been calculated by the machine in a previous part of the programme, but whether this is the case or not, it is convenient to arrange that unity is carried over into the sign digit when θ reaches the value π. It will be seen that the sign of the function can then be immediately derived from the apparent sign of the argument since when $0 < \theta < \pi$, $\sin\theta > 0$ and the sign digit of the argument is zero, but when $\pi < \theta < 2\pi$, $\sin\theta < 0$ and the sign digit is unity. Larger values of θ will also be given correctly since multiples of 2π will cause a carry past the sign digit which will be lost. Negative values are also covered for, since negative numbers are represented by complements, modulo 2, $(-\theta)$ will appear as $2\pi - \theta$, and $\sin(2\pi - \theta) = -\sin(\theta) = \sin(-\theta)$.

The sign of the function is thus determined by inspection of the sign of the argument but it is still necessary to decide whether θ (or $(\theta - \pi)$ if $\theta > \pi$) lies within the first or second quadrant. This is done by multiplying θ by 2 (one left shift) and discriminating on

the sign digit which will be 0 if $\theta<\pi/2$ and 1 if $\theta>\pi/2$. It is thus possible to decide whether θ or $\pi-\theta$ should be used in the subsequent interpolation.

The process of discrimination complete, it is now possible to consider the actual interpolation. Suppose that the discriminating process has been applied to θ, and that as a result the value $\bar{\theta}(=\theta, \pm(\pi-\theta)$ or $2\pi-\theta)$, which will be used in the interpolation, has been obtained. Suppose, further, that $\bar{\theta}=\varphi+\delta$, where φ is the argument of the next smaller tabulated value to $\bar{\theta}$. Sin $(\bar{\theta})$ is now given by the relationships,

$$\sin\bar{\theta}=\sin\varphi\cos\delta+\sin(\pi/2-\varphi)\sin\delta,$$

$$\cos\delta=1-\frac{\delta^2}{2}+\ldots$$

$$\sin\delta=\delta-\frac{\delta^3}{3!}+\ldots$$

Any desired accuracy can be obtained either by increasing the number of entries in the sine table, or by extending the number of terms in the series for cos δ and sin δ. The former alternative is preferable from the programmer's point of view since it involves only a trivial change in the code. The programme given below was constructed originally for use in calculating structure factors in connection with x-ray crystal analysis, and as only limited accuracy is required in this context, the table of sines was assumed to contain 16 entries and two terms only were taken in the expansions for cos δ and sin δ. With these limitations results will be obtained correct to 5 decimal places. The programme is as follows:

Calculation of sin θ. Range unrestricted.

Add m_8 to all memory locations.

Location	Contents	Remarks
0	$+(40)\rightarrow cA(1)$	θ
1	$Cc\geqslant 0(3)<0(2)$	
2	$+(38)\rightarrow cR(4)$	Modifies order (35) determining the sign of sin θ
3	$+(37)\rightarrow cR(4)$	Modifies order (35) determining the sign of sin θ
4	$R\rightarrow(35)(5)$	
5	$l(1)(6)$	Sign digit removed
6	$Cc\geqslant 0(9)<0(7)$	
7	$A\rightarrow(40)(8)$	

Location	Contents	Remarks
8	$-(40)\rightarrow cA(9)$	
9	$A(1\text{–}5)\rightarrow(41)(10)$	These digits specify the nearest tabulated value of θ
10	$A(1\text{–}5)\rightarrow(24)(11)$	The look-up order for sin φ is completed
11	$r(27)(12)$	The remainder δ is generated
12	$(45)\times R\rightarrow cA(13)$	δ converted to radians
13	$A\rightarrow(42)(14)$	δ is stored
14	$r(32)(15)$	δ placed in register
15	$+(39)\rightarrow cA(16)$	1
16	$-(41)\rightarrow A(17)$	$1-\varphi$
17	$A(1\text{–}5)\rightarrow(32)(18)$	The look-up order for cos $\varphi=$ sin $(\pi/2-\varphi)$ is completed
18	$(42)\times R\rightarrow cA(19)$	δ^2
19	$r(1)(20)$	$\delta^2/2$
20	$A\rightarrow(40)(21)$	$\delta^2/2$ stored
21	$+(39)\rightarrow cA(22)$	1
22	$-(40)\rightarrow A(23)$	$1-\delta^2/2\simeq\cos\delta$
23	$r(32)(24)$	
24	$(\sin\varphi)\times R\rightarrow cA(25)$	$\sin\varphi\cos\delta$
25	$A\rightarrow(43)(26)$	
26	$+(40)\rightarrow cR(27)$	$\delta^2/2$
27	$(44)\times R\rightarrow cA(28)$	$-\delta^2/6$
28	$r(32)(29)$	
29	$(42)\times R\rightarrow cA(30)$	$-\delta^3/6$
30	$+(42)\rightarrow A(31)$	$\delta-\delta^3/6\simeq\sin\delta$
31	$r(32)(32)$	
32	$(\cos\varphi)\times R\rightarrow cA(33)$	$\cos\varphi\sin\delta$
33	$+(43)\rightarrow A(34)$	$\sin\varphi\cos\delta+\cos\varphi\sin\delta=\sin(\varphi+\delta)$
34	$A\rightarrow(43)(35)$	
35	$\pm(43)\rightarrow cA(36)$	$\sin\theta$
36	$C\rightarrow(\ \ )$	
37	$+(43)\rightarrow cA(36)$	
38	$-(43)\rightarrow cA(36)$	
39	1·00–00	
40	θ	Initially. Temporary storage for $\delta^2/2$ during calculation
41	—	Cleared to zero when the programme is first inserted
42	δ	

Location	Contents	Remarks
43	—	Temporary storage
44	$-1/3$	$(1{\cdot}1010\ldots)$
45	$\pi/32$	

The final result appears in the accumulator; θ is the only parameter to be inserted, and the output order (m_8+36) must be inserted.

From the second example of interpolation considered above it is evident that use may be made of any knowledge of the analytic properties of the tabulated function. In general, when a function defined in some range is to be calculated from a limited number of terms of a power series, it emerges that the Taylor series does not, in fact, give the best representation of the function obtainable with this amount of calculation.

Several alternative methods of approach lead to better formulae in given situations. Thus, suppose that a polynomial expansion, of degree n, is required to produce the least mean square error in a range $x_e \leqslant x \leqslant x_u$, between the function $\varphi(x)$ and the approximating function. It follows that:

$$\frac{\partial}{\partial a_s}\int_{x_e}^{x_u}\left[\varphi(x)-\sum_{r=0}^{n}a_r x^r\right]^2 dx=0 \quad (r,\ s=0,\ 1\ \ldots\ n)$$

It will be seen that this set of integral relations reduces to:

$$\int_{x_e}^{x_u} x^s\left[\varphi(x)-\sum_{r=0}^{n}a_r x^r\right]dx=0 \quad (r,\ s=0,\ 1\ \ldots\ n)$$

or

$$\sum_{r=0}^{n}\frac{a_r}{r+s+1}\left[x_u^{r+s+1}-x_e^{r+s+1}\right]=\int_{x_e}^{x_u}x^s\varphi(x)\,dx \quad (s=0,\ 1\ \ldots\ n)$$

which can be solved to give the values of the coefficients a_r, when $\varphi(x)$ is a known function.

Fortunately it is not necessary to proceed in this manner, since it can be shown that the Legendre polynomials, defined by:

$$P_n(x)=\frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n$$

$$P_{n+1}(x)=[(2n+1)xP_n(x)-nP_{n-1}(x)]/n+1$$

with orthogonality relations:

$$\int_{-1}^{1} P_n(x)P_m(x)dx = 2\delta_{mn}/2n+1$$

are such that, if any function is expanded in a series of the polynomials, the first n terms of this series will give the best nth degree polynomial approximation to the given function, in the least square sense.

Whence, to obtain the best least square polynomial approximation, of degree n, to $\varphi(x)$, in the interval $-1 \leqslant x \leqslant 1$:

$$\varphi(x) = \sum_{r=0}^{n} a_r P_r(x)$$

where

$$a_r = \frac{2r+1}{2}\int_{-1}^{1} \varphi(x)P_r(x)dx$$

For any other finite interval, it is of course trivial to transform the given relationship.

Table 16.1. Legendre and Chebyshev polynomials

n	$P_n(x)$	$T_n(x)$
0	1	1
1	x	x
2	$\frac{1}{2}$ $(3x^2-1)$	$2x^2-1$
3	$\frac{1}{2}$ $(5x^3-3x)$	$4x^3-3x$
4	$\frac{1}{8}$ $(35x^4-30x^2+3)$	$8x^4-8x^2+1$
5	$\frac{1}{8}$ $(63x^5-70x^3+15x)$	$16x^5-20x^3+5x$
6	$\frac{1}{16}$ $(231x^6-315x^4+105x^2-5)$	$32x^6-48x^4+18x^2-1$

On the other hand, it may be required to find an nth degree polynomial approximation to $\varphi(x)$, such that the maximum difference between function and polynomial is minimized. In this event the relevant expansion can be shown to involve the Chebyshev polynomials, defined by:

$$T_n(x) = \cos\,(n\cos^{-1}x)$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

with orthogonality relations:

$$\int_{-1}^{1} \frac{T_n(x)\,T_m(x)}{\sqrt{1-x^2}}dx = \frac{\pi}{2}[\delta_{mn}+\delta_{on}]$$

and, for a given approximation:

$$\varphi(x) = \sum_{r=0}^{n} c_r T_r(x)$$

where:

$$c_r = \frac{2}{\pi}[1 - \tfrac{1}{2}\delta_{or}] \int_{-1}^{1} \frac{\varphi(x)\, T_r(x)}{\sqrt{1-x^2}}\, dx$$

The first few values of the Legendre and Chebyshev polynomials are given in *Table 16.1*

It is interesting to compare the coefficients resulting from the Taylor series with those obtained by polynomial expansion, and *Table 16.2* gives the values for the cosine function.

Table 16.2. Polynomial expansions of $\cos\left(\frac{\pi}{2}x\right)$

Power		x^0	x^2	x^4	x^6
Taylor		1·000 000	−1·233 701	+0·253 670	−0·020 863
Legendre	(1)	0·636 620			
,,	(2)	0·980 161	−1·030 626		
,,	(3)	0·999 578	−1·224 796	+0·226 532	
,,	(4)	0·999 993	−1·233 527	+0·252 724	−0·019 207
Chebyshev	(1)	0·472 001			
,,	(2)	0·971 405	−0·998 808		
,,	(3)	0·999 397	−1·222 744	+0·223 936	
,,	(4)	0·999 993	−1·233 472	+0·252 544	−0·019 072

As a point of analytical interest, it may be observed that the Chebyshev series for the trigonometrical functions are:

$$\sin\left(\frac{\pi}{2}x\right) = 2\sum_{r=0}^{\infty} (-)^r \mathcal{J}_{2r+1}\left(\frac{\pi}{2}\right) \cdot T_{2r+1}(x)$$

$$\cos\left(\frac{\pi}{2}x\right) = \mathcal{J}_0\left(\frac{\pi}{2}\right) + 2\sum_{r=1}^{\infty} (-)^r \mathcal{J}_{2r}\left(\frac{\pi}{2}\right) \cdot T_{2r}(x)$$

Naturally, if a single value of a particular function is to be calculated from a limited number of polynomial terms, the Chebyshev series is to be preferred. If a number of such function values are to be combined, however, the Legendre series will be used.

The virtue of the method will be apparent when it is remarked that the Legendre series, of order 6, gives nearly 100 times the accuracy

for the cosine function than is attainable with the same number of terms of the Taylor expansion. In the case of the function log $(1+x)$, E. T. Goodwin has pointed out that the Chebyshev expansion requires 14 terms to give an accuracy of 10 decimal places in the range $0 \leqslant x \leqslant 1$ whereas 10^{10} terms of the Taylor series would be needed.

OPTIMUM CODING

To conclude this section on coding, the first of the preceding examples will be re-arranged in optimum coding. First, however, it is necessary to discuss in more detail the time of execution of the various orders on *A.P.E.*(*X.*)*C.*

With the exception of the orders:

$$(x) \times R \rightarrow cA(y)$$
$$r(n)(y)$$
$$l(n)(y)$$
$$\text{Print } (y)$$
$$C \rightarrow (\quad)$$

all orders are completed before the memory location next to that referred to in the order is reached. The following order can therefore be stored in the next memory location. Thus a typical addition order might be:

$$+(26) \rightarrow A(27)$$

Multiplication is, in general, a slow process, occupying as many complete drum revolutions as there are unit digits in the multiplier. The most advantageous disposition is obtained by leaving *one* memory space before the next order (to allow for control operations) thus:

$$(26) \times R \rightarrow cA(28)$$

The shift orders $l(n)$ and $r(n)$ do not themselves refer to memory locations, and it is therefore necessary to allow for their time of execution. One location must be left for $r(n)$ and two for $l(n)$. Thus two consecutive orders might be:

$$+(26) \rightarrow cA(27)$$
$$r(n)(29)$$

or

$$+(26) \rightarrow cA(27)$$
$$l(n)(30)$$

Printing is such a slow operation compared with the other functions of the machine that optimum coding need not be considered, as the maximum time which may be wasted is negligible compared with the actual printing time.

In general it will not be possible to use optimum coding when switching from one drum to another, since this will normally be done when transferring from one subroutine to another, and the positions of these subroutines are pre-determined. In any case it is not practicable to synchronize the drums with each other and optimum coding is therefore impossible when using the order $C\rightarrow(\ )$.

One of the most important considerations, from the point of view of optimum coding, is the time taken in switching from one memory 'track' to another. It will be remembered that each memory unit of *A.P.E.*(*X.*)*C.* contains 32 tracks, each of which accommodates 16 complete 'words.' The switching time from one track to another is less than 1/16th of the time of revolution of the drum, and it is therefore sufficient to allow one memory space when switching between tracks. Thus two orders might be:

$$+(4)\rightarrow A(5)$$
$$A\rightarrow(23)(24)$$

It is useful, in this connection, to refer to *Table 16.3* which gives the relative positions of the memory locations.

DIVISION SUBROUTINE

It is now possible to give the optimally coded form of the programme of p. 162 :

Division. Forms a/m. Optimum programmed form.

Add m_9 $(=k\times 16)$ to all memory locations.

Location	Contents	Remarks
4	$(5)\rightarrow cR(6)$	Iteration index placed in register
6	$+(24)\rightarrow cA(10)$	m (Track transfer prevents $+(23)\rightarrow cA(8)$)
10	$Cc\geqslant 0(11)<0(13)$	
13	$+(31)\rightarrow cA(1)$	0 placed in position (17) if m negative
11	$+(12)\rightarrow cA(1)$	1 placed in position (17) if m positive
1	$A\rightarrow(17)(3)$	
3	$+(7)\rightarrow cA(16)$	a
16	$+(17)\rightarrow A(18)$	Sign of a changed if m is positive

Table 16.3

MEMORY LOCATION	TRACK NUMBER																															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320	336	352	368	384	400	416	432	448	464	480	496
	1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241	257	273	289	305	321	337	353	369	385	401	417	433	449	465	481	497
	2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242	258	274	290	306	322	338	354	370	386	402	418	434	450	466	482	498
	3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243	259	275	291	307	323	339	355	371	387	403	419	435	451	467	483	499
	4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244	260	276	292	308	324	340	356	372	388	404	420	436	452	468	484	500
	5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245	261	277	293	309	325	341	357	373	389	405	421	437	453	469	485	501
	6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246	262	278	294	310	326	342	358	374	390	406	422	438	454	470	486	502
	7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247	263	279	295	311	327	343	359	375	391	407	423	439	455	471	487	503
	8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248	264	280	296	312	328	344	360	376	392	408	424	440	456	472	488	504
	9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249	265	281	297	313	329	345	361	377	393	409	425	441	457	473	489	505
	10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250	266	282	298	314	330	346	362	378	394	410	426	442	458	474	490	506
	11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251	267	283	299	315	331	347	363	379	395	411	427	443	459	475	491	507
	12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252	268	284	300	316	332	348	364	380	396	412	428	444	460	476	492	508
	13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253	269	285	301	317	333	349	365	381	397	413	429	445	461	477	493	509
	14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254	270	286	302	318	334	350	366	382	398	414	430	446	462	478	494	510
	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255	271	287	303	319	335	351	367	383	399	415	431	447	463	479	495	511

Location	Contents	Remarks
18	$Cc \geqslant 0(20) < 0(19)$	
19	$l(1)(22)$	Quotient digit 1 placed in register and remainder doubled
22	$-(24) \rightarrow A(25)$	m subtracted from remainder
20	$l(1)(23)$	Quotient digit 0 placed in register and remainder doubled
23	$+(24) \rightarrow A(25)$	m added to remainder
25	$l(32)(28)$	Iteration index shifted from register to accumulator
28	$Cc \geqslant 0(29) < 0(30)$	Iteration index examined
29	$l(32)(16)$	Iteration index returned to register and remainder to accumulator. Next cycle started
30	$(31) \rightarrow cR(21)$	Register cleared to avoid possible transfer to accumulator in (21)
21	$l(1)(26)$	Iteration index removed
26	$+(27) \rightarrow A(2)$	Sign adjusted and quotient rounded off
2	$C \rightarrow (\ )$	Control transferred to main programme
17		Temporary storage for correction digit to $A(1)$
24	m	
7	a	
5	2^{-31}	
31	0	
12	1	
27	$1 + 2^{-31}$	i.e. 1·00 . . . 01

The important point which is illustrated by this programme is the necessity of paying special attention to any iteration loop which is executed a number of times. Thus the main cycle, which is repeated 31 times (locations 16 to 29 inclusive), has been so coded as to be completed within a single drum revolution, the less important start and finish sequences (4 to 3 and 30 to 2) are required only once each and are of less moment.

It should be emphasized that this is not the only possible arrangement of this programme, and it is not necessarily the best from the viewpoint of operating time. The determination of the best form of coding in cases where there are certain fixed locations which must

be referred to more than once is an extremely complex process. Moreover it suffers the disadvantage that no account is taken of compactness of arrangement so that numbers and orders for a particular problem may be widely scattered through the memory. This may cause considerable difficulty when several programmes have to be fitted in together. The example above was therefore constructed with the two objects of speed and compactness in mind.

It is now possible to compare the times required for the two forms of programme. Taking as the basic unit of time the time of one drum revolution, and assuming that a 32 binary digit quotient is required, the approximate times for the two programmes are:

Non-optimum coding	132 units
Optimum coding	35 units

showing that a saving of time of approximately 75 per cent has been achieved by the use of optimum coding.

The actual time taken by this example is only a few per cent greater than that which would result if the store had immediate access to any position. This is, however, somewhat fortuitous since the programme contains no sequences in which the same memory location is referred to twice in rapid succession. In this example the main point at which this could occur is in the increase, examination, and restoration of the iteration index; the trick of storing the latter in the register is responsible for the excellent performance and is typical of the care which must be taken to obtain the best results with a two-address code.

17

SOME APPLICATIONS OF COMPUTING MACHINES

PRELIMINARY REMARKS

THE reader, from his perusal of the preceding chapters, will be familiar not only with the way in which a modern automatic digital calculator works, but also with some of the purely mathematical techniques which are applicable to such machines. It is evident that any problem whose solution is reducible to a set of arithmetic operations can, in principle, be solved on a digital calculator. In this chapter it is proposed to examine only one problem of an immediately mathematical nature and then, in order to exhibit the scope of digital calculators, to consider briefly some other applications to situations whose relation to numerical mathematics is by no means obvious.

X-RAY CRYSTAL STRUCTURE ANALYSIS

The problem of obtaining a so-called electron density map for a crystalline substance from the experimentally observed x-ray spectra is one which has, for over twenty years, exercised the ingenuity both of mathematicians and of machine designers. The former have striven, by analytical tricks, to come more easily to the solution and the latter have sought to devise analogue equipment to lighten the burden of the crystallographer more interested in results than in techniques.

Briefly, the problem may be stated as follows; quantities $|F_0(h, k, l)|$ may be derived from the x-ray diffraction spectra of a crystal. A crystal is known to be composed of a vast number of identical building blocks or unit cells. The density of electrons, $\varrho(x, y, z)$ at any point in this unit cell is given by:

$$\varrho(x, y, z) = \frac{1}{V} \sum_{h=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} |F_0(h, k, l)| \cos(\theta - \alpha_{hkl}) \quad \ldots . \quad (1)$$

where V is the volume (in suitable units) of the unit cell,

$$\theta = 2\pi\left\{h\frac{x}{a} + k\frac{y}{b} + l\frac{z}{c}\right\} \quad \ldots . \quad (2)$$

(a, b, c) being the lengths of the cell edges. h, k, l are integers and it will be realized that the limits $\pm\infty$ applied to the above summation are replaced, in practice, by finite integers governed by the range of available experimental data. It is required to produce a set of 'maps' showing the way in which ϱ varies with x, y, z.

At this stage, it is necessary to mention the complications. In the first place the quantities α_{hkl} (phase angles) are initially unknown and are not, in general, derivable from experiment. Secondly ϱ is a function of three variables so that the best pictorial representation which can be hoped for is a set of contour maps at different levels of z, say. Finally, the information often required by the crystallographer is not a pictorial map at all, but consists of a set of detailed and accurate values of the co-ordinates of the centres

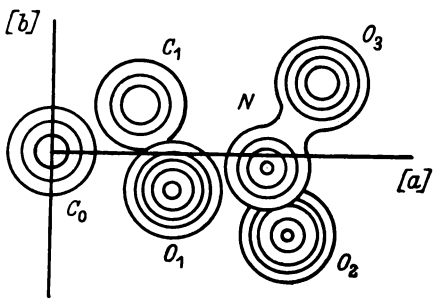


Figure 17.1

of the atoms filling the unit cell, and coinciding with the maxima of $\varrho(x, y, z)$.

To illustrate these points, *Figure 17.1* gives a view of the molecule of penta-erythritol tetranitrate $C-(CH_2ONO_2)_4$ in the form of superimposed contour maps viewed normal to the x, y plane. The superposition is necessary since all of the atoms are not at the same z level, as will be seen in *Table 17.1.*

The contours of *Figure 17.1* are drawn at intervals of unit electron density and the process of deriving them is another example of a simple though tedious calculation. The method of approach is to calculate the sum of the series (1) at the required value of z (e.g. $z=0{\cdot}124$ for the C_1 of our example), for a fixed value of y and for a range of values of x in the interval $(0\leqslant x<a)$. In practical hand calculation y is usually taken to have successively the values $(0, b/60, 2b/60 \ldots 59b/60)$, and the sums for each of these values and all values of x of the same type (i.e. $0, a/60, 2a/60 \ldots 59a/60$) are calculated. It will be seen that this process results in data which

can be used to graph the variation of ϱ with x, at a fixed z value and along lines of constant y. The intersections of these graphs with the lines $\varrho = 0, 1, 2 \ldots$ electrons/$A.U.$[3] then give the points at which the contours cross the lines of constant y, and from this data maps, of the type shown above, can be drawn. It should be noticed that the assumption has been made that complete information is available regarding the phase angles α_{hkl}.

Table 17.1. Atomic co-ordinates for penta-erythritol tetranitrate

Atom	z/a
C_0	0·000
C_1	0·124
O_1	0·256
N_1	0·359
O_2	0·494
O_3	0·326

To see how a digital computer can help with this problem it is first necessary to consider certain estimates of magnitude. Usually the number of $|F_0(hkl)|$ available is less than 1,000; individual accuracy is seldom better than 1 per cent so that they may be taken to lie in the range 0 . . . 99, i.e. to require two decimal digits for their specification, or in binary, say, 7 digits. (This will allow them the slightly greater range 0 . . . 127.) It follows that four such quantities can be stored in a single 32 binary digit word in a computer. Furthermore, noticing that α_{hkl} is often (i.e. in most structures investigated to date) 0° or 180°, and can therefore be removed from the cosine term of (1) and placed as a $\pm$ sign associated with $|F_0(hkl)|$, it is seen that both amplitude and phase of four of these quantities can be stored in the single 32 binary digit word. Since the normal storage capacity of a modern computer is at least 1,000 words, this shows that the normal quantity of experimental data is well within the internal capacity of such machines.

It is a trivial exercise in coding to separate out the constituent parts (the individual $\pm|F_0(hkl)|$ values) of each word and to use them to sum the series (1) at the required points. This, however, would lead, for each z value, to a set of $60 \times 60 = 3{,}600$ numerical values of $\varrho(x, y, z)$ from which the operator would have to draw graphs and construct a contour map in the manner previously described. Notice also that the mere output on a teleprinter of 3,600 three decimal place numbers generated by the computer

would require a time of the order of 30 min. Fortunately better methods are available, either using the standard output printer or one of the store content display devices fitted to almost all machines.

The first modification which is required in the original scheme is either the reduction of the number of lines of y to 32, or its increase to 64; for simplicity consider the former number. The second restriction which will be needed, in the simplest case at least, is that the accuracy of the x co-ordinate at which any contour crosses a line of constant y is restricted to one part in 32 (or 64). All that is now necessary is for the machine, having computed the values of $\varrho(x, y, z)$ for constant z and along any line of fixed y, to evaluate to the nearest 32nd, the x co-ordinates of $\varrho = 0, 1, \ldots$; these values, which lie in the range 0 . . . 31 32nds can now be represented by units placed correctly in an otherwise zero 32 binary digit word. When all 32 of these 'contour words' (one to each value of y) have

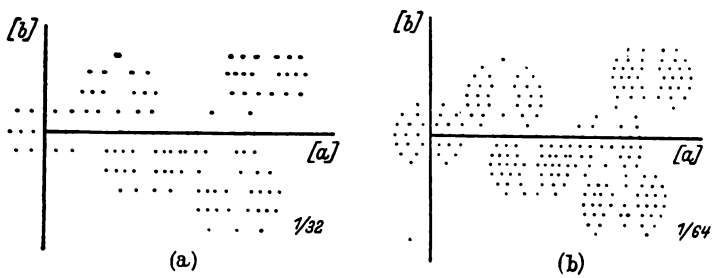


Figure 17.2 Teleprinter 'Plot' for structure of Figure 17.1

been calculated, they are simply printed, one underneath the other, by the ordinary teleprinter output the only deviation from normal being that the machine is so arranged that it spaces, without marking the paper, for all zeros, and prints a dot for the units which represent the intersections of the various contours with the y line being typed. To obtain the desired map the operator has now merely to join up with pencil lines those dots on the same contour.

The typing time for a 32×32 map of the kind just described is about 2 min, but even this can be eliminated if the storage device of the machine being used is provided with one of the normal cathode ray display devices. These are usually such as to display the digits of 32 words on successive lines of a television type raster. In *Figure 17.2* is shown a typical pattern which, in fact, is the machine output for the penta-erythritol tetranitrate projection of *Figure 17.1*. In view of the coarseness of the approximation the representation is

surprisingly good, but if greater precision is needed various means are available for producing it. A simple method is to increase the number of lines of constant y to 64 and to obtain an equal precision in x by using two words of 32 binary digits to give the x intersections. Even with this extended scope the number of storage positions required for the display is only 128; add to this the 250 positions required for the storage of the $\pm|F_0(h, k, l)|$ data, and it is seen that, with a total capacity of 1,024 words, 646 words remain for the programme, which, in fact, requires less than 300 words, and for auxiliary number locations. Even greater accuracy of mapping can be obtained if slight modifications to the teleprinter or cathode ray tube display are permitted. It is only necessary to decrease the horizontal and vertical scan rates of the latter so that the whole tube face can contain, say, 4×128 words to attain a precision of contouring better than one part in 200, and deviations of this magnitude would be invisible to the eye on a 6 in. dia. screen. A similar procedure is applicable to the teleprinter, except that here the picture produced simply occupies a greater space upon the paper.

Mention has already been made of the fact that the values of α_{hkl} are not available experimentally and, in fact, will not be available until the very end of a structure determination. Maps of the type described above are seldom, if ever, required to an accuracy of better than 1 per cent, and a process will now be described which enables a computer to carry on the refinement of a structure from the position in which the $|F_0(h, k, l)|$ are known from experiment and also the approximate locations of the atoms in the unit cell. This situation is, indeed, the one at which the crystallographer is faced with the mass of numerical work envisaged in the earlier part of this section.

It can be shown that, if $(x_r, y_r, z_r)[r = 1 \ldots n]$ are the co-ordinates of the atomic centres in the unit cell,

$$|F_c(h, k, l)| = \sum_{r=1}^{n} f_r \cos(\theta_r - \bar{\alpha}_{hkl}) \quad \ldots . \quad (3)$$

$$\tan \bar{\alpha}_{hkl} = \sum_{r=1}^{n} f_r \sin \theta_r \Big/ \sum_{r=1}^{n} f_r \cos \theta_r, \quad \ldots . \quad (4)$$

where

$$\theta_r = 2\pi\left\{h\frac{x_r}{a} + k\frac{y_r}{b} + l\frac{z_r}{c}\right\}$$

and f_r is a quantity known as the 'atomic scattering factor' whose value has been calculated for all atoms. $|F_c(hkl)|$ is a quantity whose value is identical with that of $|F_0(hkl)|$ when the $(x_r y_r z_r)$

coincide with the atomic positions of the atoms in the real crystal, and, in a similar manner $\bar{\alpha}_{hkl}$ coincides in value with α_{hkl}.

The reader will now see that an iterative refinement process can be applied to give accurate values of $(x_r y_r z_r)$ once an approximate set is known. The cycle of operations is as follows:

(a) Calculate $\bar{\alpha}_{hkl}$ from the assumed atomic co-ordinates $(x_r y_r z_r)\{r = 1 \ldots n\}$.

(b) Insert $\bar{\alpha}_{hkl}$ in place of α_{hkl} in series (1).

(c) Sum the series with $|F_0(hkl)|$ and $\bar{\alpha}_{hkl}$.

(d) Find the resulting atomic co-ordinates by observing that they correspond to maxima of $\varrho(x, y, z)$ and that at its maxima: $\frac{\partial \varrho}{\partial x} = \frac{\partial \varrho}{\partial y} = \frac{\partial \varrho}{\partial z} = 0$, call these maxima $({}_1x_r, {}_1y_r, {}_1z_r)$ *etc.*

(e) Compute $\sum_{r=1}^{n} (x_r - {}_1x_r)^2 + (y_r - {}_1y_r)^2 + (z_r - {}_1z_r)^2 = D$, say.

(f) If $D > 0$ to the accuracy of $(x_r y_r z_r)$ return to (a) and repeat the above process using $({}_1x_r, {}_1y_r, {}_1z_r)\{r = 1 \ldots n\}$ in place of the original (x_r, y_r, z_r).

If $D = 0$ print the resulting $({}_1x_r, {}_1y_r, {}_1z_r)\{r = 1 \ldots n\}$ which represent the required parameters.

It is not possible here to enter more deeply into the numerical processes involved in the above cyclic refinement technique, but it may be mentioned that, for the substance penta-erythritol tetranitrate discussed above, the application of the method by hand required about 12 months' work, whereas today a high-speed machine could attain the same result in as many minutes.

As a final example of a machine application to crystallography there is the solution of the above problem by a minimization process. Such methods are widely applicable to many problems both of pure and of applied science and are likely to receive extensive use as more high-speed machines become available.

For the crystallographic example the process depends upon the fact that as the trial co-ordinates $(x_r, y_r, z_r)[r = 1 \ldots n]$ approach more and more nearly the true co-ordinates, the values of $|F_0(hkl)|$ and $|F_c(hkl)|$ will tend to equality. This may be expressed mathematically in the statement that:

$$R = \sum_h \sum_k \sum_l (|F_0(hkl)| - |F_c(hkl)|)^2 \qquad \ldots . \quad (5)$$

tends to a minimum, and it may be shown that if (x_r, y_r, z_r) $[r = 1 \ldots n]$ are trial co-ordinates giving a particular value of R

(say R_0), then a lower value of R is produced by the co-ordinates: $(x_r+\varepsilon_{r1})(y_r+\varepsilon_{r2})(z_r+\varepsilon_{r3})$ where:

$$\varepsilon_{rj}=\frac{-[\sum_{rj}(\partial R/\partial x_{rj})^2].\partial R/\partial x_{rj}}{\sum_{rj,r'j'}(\partial^2 R/\partial x_{rj}\partial x_{r'j'})\,.\,\partial R/\partial x_{rj}\,.\,\partial R/\partial x_{r'j'}} \quad \ldots . \quad (6)$$

and, for simplicity $x_{r1}=x_r$, $x_{r2}=y_r$, $x_{r3}=z_r$.

This method can again be incorporated into a cyclic process in the following way:

(a) Calculate all $\varepsilon_{rj}\{r=1\ldots n;\ j=1\ldots 3\}$ from trial co-ordinates $(x_r y_r z_r)\{r=1\ldots n\}$.

(b) Compute $\sum_{rj}\varepsilon^2_{rj}=D$, say.

(c) If $D>0$ return to (a) and repeat the above process using ${}_1x_r=x_r+\varepsilon_{1r}$, *etc.*

If $D=0$ print the original $(x_r y_r z_r)\{r=1\ldots n\}$ which are the required co-ordinates.

It will be seen that, conceptually at least, this process is considerably simpler than that originally described, although the expressions (6) are not so elementary in character as the straightforward trigonometrical summations involved in (1) and (4).

MECHANICAL TRANSLATION

An interesting example of a computing machine application which, at first sight, seems non-numerical is the translation from a foreign language into some base language. That this would be possible with almost any modern computer, was first suggested by the authors in 1947, and since that date considerable progress has been made both semantically and from the engineering points of view.

In the first place it must be stated, quite clearly, that what is envisaged, by the present authors at least, is a dictionary translation with possible grammatical notes. It is in no way expected that a translation of literary quality will be produced.

The original idea was to set up, in the store of the machine, a dictionary of foreign language (*F.L.*) words and their equivalents in the base or target language (*T.L.*). Mechanical translation (*M.T.*) would then consist of supplying the message to be translated to the machine *via* a teleprinter, one word at a time. Input words would then be compared with the *F.L.* entries and the *T.L.* equivalent (or equivalents) would be typed out. This process is obviously unsatisfactory in a number of particulars; for example, a dictionary of

enormous size would be required and even this would be unlikely to contain enough variants to interpret a simple message.

A greatly improved version of the process was suggested by R. H. Richens; this depends upon the observation that, in translating scientific prose at least, a vocabulary limited to about 1,000 words of technical jargon, specific to the field of reference of the paper, together with a further 1,000–5,000 words of common usage, will provide an adequate translation of most material of this type. To overcome the semantic problems of multiple words (e.g. German) and of syntax, it was proposed that the dictionary should consist of two parts, the first containing *stems* with their translations, and the second containing endings with suitable syntactic notes.

The process of translation is then as follows. Each word of the *F.L.* text is compared with the stem dictionary, the longest stem common to the *F.L.* word and to the dictionary is translated by the latter and printed at the output. Next the remainder of the *F.L.* word is compared with the ending dictionary and this time syntactic information is typed out. This process is repeated for all words in the *F.L.* text. An example given by Richens is the following:

		Stem	Ending	
F.L.	*amo*	*am*	*o*	
T.L.		love (*v*)	1*sp*	(1*sp* = 1st singular present tense) (*v* = verb)

Here the translator observes that although the short stem *a* (=alas) occurs in the stem dictionary the longer stem *am* is also present but that the complete *amo* does not appear. The stem translation 'love' with the verb indication (*v*) is therefore given and the residue *o* is compared with the ending dictionary where it is found to indicate the first person singular of the present tense (1*sp*). Naturally this simple scheme requires some modification in order to take account of multiple words of the type common in German, and also of ambiguities. Examples taken from Richens are: the three entries in the stem dictionary which would be needed for *rego* —I rule, *viz. reg*, *rex* and *rect*; the ambiguity which arises when the word *desideremus* is to be treated, *viz.*:

desider = desire	*emus*	(1 *p.p.s.a.*)
desid = be idle	eremus	(1 *p.i.s.a.*)

which could only be resolved by including the whole word and its two translations. Again, certain words (e.g. the French *t* as in *a-t-il*) are of no semantic significance and will not be translated;

to assure the user of the machine that no error has occurred it is well to output some signal such as (*v*) (=vacuous).

Enough has now been said to enable the reader to see the workings of the method, and the problem of applying an automatic digital calculator to (*M.T.*) will now be considered. The most obvious method is to code each letter of the dictionary word in 5 binary digit form and to store the translation in that memory location having the same digital value as the aggregate value of the dictionary word. Translation would then consist in coding the message word (a teletyper does this automatically) and then extracting the translation in the same or next lower digitally valued storage position. The first case gives an exact translation and the second a stem translation with a remainder. The stem translation would be at once typed at the output station and the undecodable remainder shifted to occupy the extreme left-hand position of the word storage register. When this had been done the number remaining would be used to refer to the ending dictionary whose entry would again be typed at the output. In event of the word forming part of a multiple semantic unit the whole word would be sent to an auxiliary storage position to await receipt of its remaining part or parts, an indication being typed meanwhile.

Despite its simplicity this scheme is quite impracticable with any existing storage device because the naïve process of placing each dictionary word in a storage position equivalent to its own binary numerical value is grossly redundant. For example, 5,000 words should require a storage capacity of only 5,000 positions, whereas if the maximum word length is 10 letters, in straight binary code any number up to 2^{50} *might* represent a word (although in fact only 5,000 or approximately 2^{12} would do so). This means that, in decimal numbers, a storage capacity of about 10^{15} words would be needed!

To overcome this difficulty a slightly more sophisticated approach has to be adopted. Dictionary words are stored in alphabetical order and in consecutive memory locations. Possibly these might be broken up into initial letter groups such that each group starts in a storage location indicated by the binary numerical value of its initial; this would increase the speed of operation by a considerable factor without greatly adding to the storage requirements. The coded word to be translated is sent to an arithmetic register and is there compared, by subtraction, with the dictionary words. In this way, using the normal conditional transfer order, the dictionary stem corresponding to the word can be obtained and its translation printed. Next the remainder of the message word, left after

subtraction of the stem, is shifted to occupy the extreme left-hand register positions and the comparison process repeated, this time with the words of the ending dictionary. This process would lead to a further typed output giving the required grammatical information, after which the machine would be ready to receive the next message word for translation. Parts of multiple semantic units would be stored for future treatment exactly as in the more elementary scheme outlined above.

To complete the discussion it is only necessary to make some remarks about times of operation of general purpose computing machines when used for this process. The relevant feature, so far as the second treatment described above is concerned, is the time of a single subtraction and this can be taken conservatively as 500 μsec. Thus, even assuming a complete dictionary search over 5,000 words to be necessary, the total translation time per word would be a maximum of 2·5 sec and a mean of 1·25 sec. To this must be added a typing time which, for 40 letters, would be about 6 sec. It would be uneconomical to reduce the translation time by a factor of more than 2 and this could be attained easily by means of the initial letter group storage technique outlined above.

Translations prepared by these techniques are curious in appearance since some words are accompanied by several meanings all of which are typed; experiment has shown, however, that the results are useful and easily read by the expert in the subject of translation. The elimination of semantic ambiguities is a problem which has received considerable attention and a recent suggestion by Reiffler is interesting. Reiffler suggests the use of a *pre*-editor who need not know the (*T.L.*) at all but who, reading his own language (our *F.L.*), eliminates from the text all such alternatives; this can be done either by the use of special marks which will enable the machine to select between alternative meanings, or by the replacement of the offending words by unambiguous synonyms. In the same way a *post*-editor might be used to turn the translation into acceptable *T.L.*

GAMES

A somewhat surprising application of automatic digital calculators is that of opponent in various games. Possibly the most elementary type of play for a machine is typified by the game of two-handed whist. In this game each player is dealt 13 cards, those in the hand of the machine are indicated to it by insertion *via* the normal input, card suits and values being given in a simple code. Fifty-two

locations in the store of the machine are set aside for card information, and when the machine receives its hand an indication is placed in each relevant position.

When play starts it will be assumed at first that the machine does not lead. As the other player puts down a card, the value of this is simultaneously supplied to the machine, which records it in the appropriate card location. When the turn of the machine to play is reached, it examines its opponent's card in the current trick, searches its card record position to see if it has any cards of the given suit, and if so plays in one of two ways,

(a) If it has cards of higher denomination than that in play it plays that card which is just greater and takes the trick.

(b) If it has only cards of lower denomination than that in play it plays the least card available.

On the other hand, the machine may have no cards of the given suit. In this event it examines its stock of trump cards and plays the lowest valued of these, thus taking the trick. Or again, if no trumps are available, the machine selects the lowest valued card available in any suit.

The play of the machine is readily indicated to its human opponent, either by its typing the denomination and suit of the card to be played, or by some form of special indicating lights.

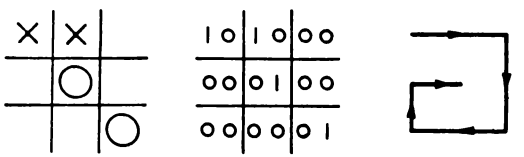
When the machine is to have first play, the situation is slightly more complicated, a reasonable strategy is to play the strongest non-trump card available to test the strength of its opponent's hand, and the coding of this procedure is quite straightforward.

More complicated card games are equally amenable to machine play and any of the normal human strategies are easily programmed.

In most card games there is an element of chance and the result cannot be predicted in advance. The game of noughts and crosses, however, is one in which the result, with skilled players at least, is not in doubt. It is a draw. Special machines have been built for playing noughts and crosses, but the game is equally well programmed on an automatic digital calculator. The programme adopted depends largely upon the available storage space, the simplest method is first to tabulate all of the possible games, observing that the ordinary square and its contents can be regarded as a ternary number. For example, let the contents of each square be represented by:

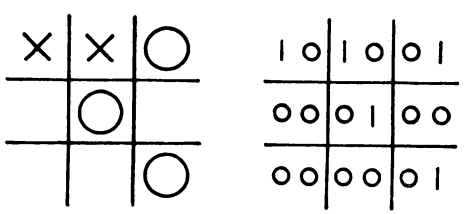
. 00 if no play has occurred
01 if occupied by a '0'
10 if occupied by a 'x'

Then the situation shown in *Figure 17.3* would be represented by the 18 digit array 101000,0001,0000,00,01. This can be regarded as the first part of a normal 32 binary digit word the remainder of which is zero. All stages of all possible games are embraced in this scheme and the technique of coding is as follows. All possible game code numbers (previously worked out) are stored in consecutive locations. Each of the stored game code numbers has 14 spare digits and 9 of these can be used to represent the optimum machine



101000, 0001, 0000, 00,01

Figure 17.3



101001, 0001, 0000, 00,01

Figure 17.4

move (the ternary numbers used above are not required since only two actions can take place, either the machine is to mark a given square or it is not to mark it). In the above example let it be assumed that the machine is to move next and that it is '0.' The above code number would then be found in the store as:

$$101000,0001,0000,00,01|00000,\underbrace{001,00,00,0,0}_{\text{machine move}}$$

and the situation after play would be as shown in *Figure 17.4.* The code number representing the state of play is fed to the machine, it is compared—by subtraction—with each of the stored code numbers starting with that of least numerical value. As soon as the result of the subtraction becomes positive (indicated by a conditional

transfer instruction) it is known, from the above discussion, that the number standing in the accumulator is simply the last digits of the stored code number which represent the machine move. It is simple at this stage, either to print by means of a suitable sub-routine the required machine move, or to display it on a set of nine neons attached to the machine accumulator. (The reader may like to convince himself that the termination of the game shown in *Figures 17.3* and *17.4* is given by:

101001,1001,0000,00,01|00000 000,00,01,0,0
or 101001,0001,0010,00,01|00000 000,10,00,0,0

both leading to a win for '0'—the machine!)

There are several sets of games which are both complete, that is contain all possible opponent's moves which follow the prescribed machine moves, and optimum in the sense of guaranteeing that the machine will never lose. One such set contains 512 games each having four machine moves and, in the case of a careless opponent, produces 360 wins. It follows that to contain all of these games the high speed store would require to have a capacity of 2,048 words to which must be added the space occupied by the programme.

To overcome this difficulty two courses of action are available, the first is to observe that the noughts and crosses square has considerable symmetry; for example, there is a four-fold rotation axis through its centre and mirror planes at angles of $n\pi/4$ through the same point. All of these elements are not independent but it can be seen that to each 'basic' game correspond seven more obtained by the symmetry operations. It follows that the above 512 games may be divided by 8 so that only $4 \times 64 = 256$ states of play have to be stored. When this is done the programme must necessarily contain a set of operations which reduce the input configuration to one of the standard forms, and likewise re-orientate the machine moves to suit the actual game. This is the reason for the spiral scanning array in which the configuration number has been formed, and from which symmetry re-orientation can be obtained by simple shift operations in a machine register. The second method of play is by the use of a standard machine strategy or set of rules of play which will never lead to a lost game. Several sets of this type are applicable to noughts and crosses, one is:

(a) *Machine* [0] *has first move*

(1) Places '0' at centre.

(2) Scans all allowable rows, columns and diagonals.

(3) If '0' has two marks in any row, column or diagonal having vacant site, place '0' there to win game.

(4) If 'x' has two marks in any row, column or diagonal having vacant site, place '0' there. (Assuming (3) does not hold.)

(5) Place '0' in corner if possible.

(6) Otherwise place '0' next to any 'x.'

(b) *Opponent* [X] *has first move*

(1) Place '0' in centre if possible.

(2) Otherwise place '0' in left-hand upper corner.

(3) Exactly as in (2) . . . (6) of previous scheme.

It is clear that a set of rules of this type lends itself to machine use and that, by their use, the storage of large numbers of standard games is avoided.

As a final example of the playing of a game by an automatic calculating machine brief mention will be made of the game of chess. Here, again, there are two principles of approach, in the first standard games and openings could be recorded in the store and typed out in any given situation. This would be quite impracticable with any machine at present in existence as the capacity is far too small; on the other hand, it is well to remember that a technique much akin to this is often used by human players skilled in the game. The second approach depends upon the existence of so-called position strength formulae. In these, by considering the position, type and environment of any piece on the board, a figure indicating its value is derived. Suppose that the game is in any stage and that the machine (black, say) is to play. The position strengths for both black and white are calculated and their ratio R evaluated. The machine now moves each of its own pieces (computationally only, of course) and examines the resulting value of R. A simple game results from then taking that move which maximizes R. This would be exceedingly bad and uninteresting play, a better state of affairs results if the machine computes the matrix of R values which is obtained by making each possible move of its own pieces and then evaluating R after each possible retaliatory move by white, the move giving either the maximum element, or the maximum $\Sigma|R|$, over all white's countermoves being then selected. This process can be continued, in theory at least, for several moves and might lead to a passable game; on the other hand it is salutory to examine the storage capacity and time requirements of the process. Very approximately it may be assumed that at any stage 10 pieces can be moved, and that only one move is possible to each

piece. Furthermore, allow the possibility of 10 additions to compute the position strength after each move (a gross under-estimate). Then the elementary game will require 100 addition times per move and storage for some 100 positions and quantities. With an addition time of 10 μsec (about the fastest in any available machine) this implies 1 msec per move, a very reasonable figure. The next order of sophistry, when white's countermoves are also considered, will require 100 extra storage positions and 1,000 addition times or 10 msec for computation, still a reasonable figure. However, it is now evident that the computing time and storage requirements increase by a factor of 10 at each level of the process so that if 17 minutes $\simeq$1,000 sec is assumed to be the longest allowable time between moves, the fastest machine now available could not hope to plan more than two or three moves ahead and this would strain the resources of any existing storage device.

It is clear that some improvement in this situation might be produced by limiting the machine computations to those paths of high initial R value, on the other hand this might lead to a failure to recognize potentially good moves of the sacrificial type.

MACHINE LEARNING AND INTELLIGENCE

To conclude this book it is proposed to give a brief account of the status of the two qualities mentioned in the heading and to discuss the extent to which they may be associated with automatic digital calculators.

In the first place, when considering the subject of learning by experience, it is necessary to distinguish between two distinct processes. The first is better described as the formation of a conditioned reflex, and is distinguished by the fact that the learner is not told the response to be learned but has to deduce it from the behaviour of the teacher. For example, a hoop is placed in front of a dog, if he jumps through it he is praised and rewarded with food, if he turns away he is beaten and re-directed towards the object. This type of learning is associated with unintelligent or uneducated subjects, and does not pre-suppose any understanding on the part of the latter of the object of the teacher's activities. The second type of learning occurs when the learner is aware of the desires of the teacher and is required to memorize or acquire a skill. Examples of this are, the learning of a multiplication table or a poem, and the execution of a pianoforte sonata.

Now it must be realized, from the start, that an arithmetical machine is limited in its response to the emission of letters or figures

at its output and that, similarly, the only awareness which it can have of conditioning or teaching activity is *via* its input. With these limitations the behaviour of an automatic digital calculator under these two types of instruction can now be investigated. The second process is the easier to consider, it corresponds to a partially educated child so that it may be assumed that the machine is capable of receiving a programme. All that is necessary is to insert first a read-out programme and secondly the numerical data to be learned and repeated; thereafter each time the machine is invited to 'recite' the table or other data—by means of an instruction to start at the first order of the read-out programme—a completely correct version will be typed. In this respect a machine is much superior to a human being, since only one insertion of data is necessary, and (assuming that it is operating correctly) the machine will never forget or make mistakes. More elaborate learning tasks of the same type may be set to the machine; for example, the problem of shopping in which certain articles, represented by code numbers, are to be obtained from various shops, again code numbers. On receipt of an article code number the machine outputs the shop code numbers in sequence. The shopkeeper (the human machine operator) indicates to the machine that the goods are in stock or otherwise; if the latter, the machine proceeds to the next shop, if the former, it records an association in such a way that whenever asked for the particular article again the same shop is immediately patronized. A slight extension will enable a further search to be made if at any future time an article becomes 'out of stock.'

The programme for this problem is as follows. Let the article be numbered $1 \ldots r \ldots n$, and the shops $1 \ldots t \ldots m$. To shop for article r record r in $(s+4)$ and then start the machine on the order in storage location r.

Location	Order	Effect
r	$+(n+1)\rightarrow cA(s+1)$	Initially the number of shop 1 is sent to the accumulator and printed. The machine then stops for approval, or otherwise
$s+1$	Print$(s+2)$	
$s+2$	Stop$(s+3)$	
		If shop number 1 has a stock of article r all is well and the operator may start on the next article
If no stock of article r re-start machine on position $s+3$ which is already awaiting a start instruction		

Location	Order	Effect
$s+3$	$+(s+4)\rightarrow cA\ (s+5)$	r
$s+5$	$A(1\text{–}9)\rightarrow(s+6)(s+7)$	r substituted in order to read $+(r)\rightarrow cA(s+15)$
$s+7$	$+(s+8)\rightarrow cA(s+9)$	
$s+9$	$+(s+4)\rightarrow A(s+10)$	r substituted in x address of order initially $A\rightarrow(0)(0)$
$s+10$	$l(23)(s+11)$	y address lined up to receive r
$s+11$	$+(s+4)\rightarrow A(s+12)$	r added to y address
$s+12$	$l(41)(s+13)$	Order returned to original digital position
$s+13$	$A\rightarrow(s+14)(s+6)$	Order returned to store
$s+6$	$+(r)\rightarrow cA(s+15)$	Shopping order
$s+15$	$+(s+16)\rightarrow A(s+14)$	2^{-p} (i.e. 1) added to shop number
$s+14$	$A\rightarrow(r)(r)$	Repeats programme from start, now trying shop number 2

Storage locations

Location	Contents
$1\ldots$	Order $+(n+1)\rightarrow cA(s+1)$ initially
$\vdots$	Order $+(n+1)\rightarrow cA(s+1)$ initially
n	Order $+(n+1)\rightarrow cA(s+1)$ initially
$n+1$	1
$n+2$	2
$\vdots$	
$n+m$	m
$n+m+1$	A number which when printed is decoded as 'no stock in any shop'
$s+4$	r
$s+6$	$+(\ \ )\rightarrow cA(s+15)$
$s+8$	$A\rightarrow(0)(0)$
$s+14$	Blank to receive $A\rightarrow(r)(r)$
$s+16$	2^{-p} which is used to increase shop number by unity

It will be noticed that if at any time the operator decides that any shop shall run out of stock of a particular item the machine, after first trying this shop, will continue its search until another supplier is found, or it has established that no supplier is available. A defect of the above simple programme lies in the fact that once a shop has been found to have no stock of a particular article it will

never be tried again. This could be easily remedied by inserting a set of instructions which would, on receipt of a no stock indication for shop r, examine the word in location r to see if it implied a trial of earlier shops at a previous time (by finding if the location was $n+1$ or some greater number) in this event, instead of proceeding to higher shop numbers it would start again at the first shop.

The problem of producing a conditioned reflex in a digital calculator is much more complicated. To be fair in the process it is not strictly permissible to insert any programme whatever into the store (corresponding to the disordered state of a newly born animal). On the other hand, animals are constructed so as to have certain inborn reflexes and these make possible their further education (for example, a desire for food and a reaction against pain). In 'educating' a machine the desired characteristics would be of the type: after insertion of the number 1, say, at the input the machine produces a random output, this is *not* the desired number, so disapproval is indicated in some way, the 1 is re-inserted and the machine again reacts. This process is continued until the machine emits the number $\pi = 3{\cdot}14$ *etc.*, say. This is probably too ideal a situation and a more realistic procedure would be to insert the digits of π after the 'punishment' and hope that the machine would realize what was required of it. Naturally, it is not permitted that a recording programme be used as this would make the problem trivial, neither is the shopping type of programme acceptable.

A possible line of approach is the following, when the machine is first switched on (corresponding to 'birth'), but not started up, its storage and register units will be filled with random numbers. (In the magnetic drum type of calculator it will be assumed that the drum is 'clean,' that is having neither 0's or 1's recorded upon its surface. In this event the output from the amplifiers will be purely random.) The conditioner applies a stimulus, for example, the instruction to start on the instruction held in storage position 1. At this stage various things may happen, for instance, the random contents may, in fact, correspond to the programme required to produce the desired output; more probably the machine will either refuse to start at all (the random order is undecodable), or else execute a few operations and then stop. The problem at this stage is to decide on an appropriate form for the approval or disapproval stimulus and here, again, the problem is not hopeless. If the machine refuses to start an appropriate 'punishment' would be to momentarily interrupt its power supply, thus changing the random contents of store and registers, and repeated use of this technique will eventually produce a result. On the other hand some activity

may have taken place and this must be greeted with 'encouragement.' If it is assumed that the machine has started and stopped without printing an output and that the sequence of operations executed is known ('observing the student'), it is possible to restart the machine at the original place and stop it again at the most undesirable operation, this could then be altered to a better variant by local shock treatment to the control register. (Again paralleled by common situations in the education of the young.) The aim of these somewhat crude processes is, of course, the setting up of genuine learning programmes inside the machine so that its memory faculty and elementary reasoning powers (e.g. shopping) could be used effectively to speed up the education, and it seems that some, perhaps more elaborate, version of the foregoing may constitute a valid reason for asserting that a machine can learn from experience.

An interesting extension of this idea arises when the response patterns of the young of the human species are considered. Certain fundamental characteristics are always present, among them the desire for food, the instinct for self preservation, and the avoidance of pain. These are often made the basis of educational technique and suggest a more appropriate means of educating a machine. It is assumed that we know the 'instincts' of the machine, namely the fact that it is designed to store numbers and do arithmetic. On the other hand we do not know how to communicate with it except through the input and output organs of the computer. The problem is then to apply different stimuli to the input and to deduce from the output, what action the machine is taking. When the machine reaction to all possible individual input stimuli has been found, it would then be easy to insert any desired programme or data for storage in the normal manner. Should the reader think that this is a rapid and trivial process, it may be pointed out that even assuming the word length of a computer to be known (which it would not be initially) for a control word of 32 binary digits nearly 10^9 input combinations would be possible and it would take the most seasoned designer some time to find, by trial, which groups of digits referred to orders, which to positions, and so on, let alone the exact details of the code. All of this is not without its implications in ordinary educational technique and it is quite possible that if the code were known the elementary instruction of the very young could be greatly speeded up.

It should be mentioned, before leaving this subject, that elementary programmes can be devised so that the actions of a machine under given stimuli will simulate an animal with any desired inbred behaviour characteristics.

Finally, consider the question—can an automatic digital calculator be considered intelligent? In the strict sense any answer to this question is a matter of taste, since intelligence would mean two entirely different things to, say, a dock labourer and a university professor. Since, within the range of its accumulated programme material, a computer is completely accurate and has perfect memory faculties most ordinary tests would indicate its superiority to almost any human. On the other hand, the present authors would prefer to define relative intelligence by relative capacity for creative work. It appears that human creative activity is of two kinds:

(1) Deduction of new conclusions from existing facts.

(2) Production of completely new ideas not deducible from known data.

In principle, any creative activity of the first kind can be achieved (though not, perhaps, in the quickest way) by an analysis of existing data by the processes of symbolic logic. Binary calculators are particularly suitable for this type of work, and there appears to be no reason to doubt that in time, such will be well within the scope of activity of these machines. Creative work of the second type is extremely rare, and it appears, on examination, that it is usually, if not always, a result of random processes. For example, at random, one may make the statement 'the moon is made of green cheese,' the un-original mind might either not have this fancy at all, or reject it as absurd without further trial. The genius, however, would consider it in detail and thus possibly discover a fundamental natural law. There appears no reason, in principle, why an automatic digital calculator, fitted with a source of random numbers, should not examine these (considering them to represent coded propositions) by the processes of mathematical logic, and in relation to all of the known facts of nature contained in its store. It would then either reject them on logical grounds, or call on its human ancillaries to investigate the subsidiary consequences of the main proposition which would either collapse or take its place as a new law of nature. The purely binary choice as to whether this constitutes machine intelligence we leave to the reader.

BIBLIOGRAPHY

BOOKS

A.1 AIKEN, H. H. (Ed.), 'A Manual for the Operation of the A.S.C.C.', *Annals 1*, Harvard, 1946

A.2 —, 'Description of a Relay Calculator, Mk. 2', *Annals XXIV*, Harvard, 1949

A.3 —, 'Proceedings of a Symposium on Large Scale Calculating Machinery', *Annals XVI*, Harvard, 1948

A.4 —, 'Proceedings of a Second Symposium', *Annals XXVI*, Harvard, 1951

A.5 —, 'Synthesis of Electronic Computing and Control Circuits', *Annals XXVII*, Harvard, 1951

A.6 BABBAGE, C., *Passages from the Life of a Philosopher*, London, 1864

A.7 BABBAGE, H. P., *Babbage's Calculating Engines*, London, 1889

A.8 BAXANDALL, D., 'Calculating Machines', *Ency. Brit.*, 14th ed., 4 (1929), 551

A.9 —, *Calculating Machines and Instruments*, London, 1926

A.10 BEGUN, S. J., *Magnetic Recording*, New York, 1948

A.11 BERKELEY, E. C., *Giant Brains*, New York, 1949

A.12 BOOLE, G., *Treatise on the Calculus of Finite Differences*, London, 1872

A.13 —, *Collected Logical Works*, Vol 2, Chicago, 1940

A.14 BOOTH, A. D., *Fourier Technique in X-ray Organic Structure Analysis*, Cambridge, 1948

A.15 BOWDEN, B. V., *Faster than Thought*, London, 1953

A.16 CAJORI, F., *A History of Mathematics*, 2nd ed., p. 485, New York, 1919

A.17 CRANK, J., *The Differential Analyser*, London, 1947

A.18 CREMER, H., *Probleme der Entwicklung*, Aachen, 1953

A.19 COUFFIGNAL, L., *Les machines à calculer*, Paris, 1933

A.20 —, *Les machines à penser*, Paris, 1952

A.21 ECKERT, W. J., *Punched Card Methods in Scientific Computing*, New York, 1940

A.22 E.R.A. Staff, *High Speed Computing Devices*, New York, 1950

A.23 FREEMAN, H., *Actuarial Mathematics*, Cambridge, 1931

A.24 GIBB, D., *Interpolation and Numerical Integration*, London, 1915

A.25 HARTREE, D. R., *Calculating Machines*, Cambridge, 1947

A.26 —, *Calculating Instruments and Machines*, Cambridge, 1949

A.27 —, *Numerical Analysis*, Oxford, 1952

A.28 HORSBURGH, E. H., *Modern Instruments of Calculation*, London, 1914

A.29 JORDAN, C., *Calculus of Finite Differences*, New York, 1947

A.30 KEISTER, W., RITCHIE, A. E., and WASHBURN, S. H., *Design of Switching Circuits*, New York, 1951

A.31 KNOTT, C. G. (Ed.), *Napier Tercentenary Memorial Volume*, London, 1915

A.32 KORFF, S. A., *Electron and Nuclear Counters*, New York, 1946

A.33 KORN, G. A., and KORN, T. M., *Electronic Analog Computers*, New York, 1952

A.34 LANCZOS, C., 'Tables of Chebyshev Polynomials', *Rep. nat. Bur. Stand., Wash.*, 1952

A.35 LEVY, H., and BAGGOTT, E. A., *Numerical Studies in Differential Equations*, Vol. 1, London, 1934

A.36 LEWIS, W. B., *Electrical Counting*, Cambridge, 1948

A.37 MENEBREA, L. F., *Scientific memoirs*, Vol. 3, London, 1842

A.38 MEYER ZUR CAPELLEN, W., *Mathematische Instrumente*, 2nd ed., Leipzig, 1944

A.39 MILNE, W. E., *Numerical Calculus*, Princeton, 1949

A.40 MILNE-THOMSON, L. M., *The Calculus of Finite Differences*, London, 1951

A.41 MURRAY, F. J., *The Theory of Mathematical Machines*, New York, 1947

A.42 D'OCAGNE, M., *Calcul Graphique et Nomographie*, Paris, 1914

A.43 SCARBOROUGH, J. B., *Numerical Mathematical Analysis*, 2nd ed., Baltimore, 1950

A.44 SHANNON, C. E., and WEAVER, W., *Mathematical Theory of Communication*, Illinois, 1949
A.45 SNOW, C., 'Hypergeometric and Legendre Functions', *Rep. nat. Bur. Stand., Wash.*, 1952
A.46 STEFFENSON, J. F., *Interpolation*, Baltimore, 1927
A.47 SVOBODA, A., *Computing Mechanisms and Linkages*, New York, 1948
A.48 WELD, C. R., *A History of the Royal Society*, Vol. 2, chap. 11, London, 1848
A.49 WHITTAKER, E. T., and ROBINSON, G., *Calculus of Observations*, London, 1944
A.50 WIENER, N., *Cybernetics*, New York, 1948
A.51 —, *Extrapolation, Interpolation and Smoothing of Stationary Time Series*, New York, 1949
A.52 —, *The Human Use of Human Beings, Cybernetics and Society*, Boston, 1950
A.53 WILKES, M. V., *et al.*, *The Preparation of Programs for an Electronic Digital Computer*, Cambridge, Mass., 1951
A.54 WILLERS, F. A., *Practical Analysis, Graphical and Numerical Methods*, New York, 1948

REPORTS, OF LIMITED CIRCULATION

B.1 BARGMANN, V., *et al.*, *Solution of Linear Systems of High Order*, Princeton, 1946
B.2 BIGELOW, J. H., *et al.*, *Interim Progress Report on the Physical Realisation of an Electronic Computing Instrument*, Princeton, 1947
B.3 —, —, *Second Progress Report*, Princeton, 1947
B.4 —, —, *First Progress Report on a Multi-channel Magnetic Drum Inner Memory*, Princeton, 1948
B.5 BOOTH, A. D., *Design for D.E.C.*, London, 1951
B.6 —, and BRITTEN, K. H. V., *General Considerations in the Design of an All Purpose Electronic Digital Computer*, Princeton, 1947
B.7 —, —, *Coding for A.R.C.*, Princeton, 1947
B.8 —, and STREET, L. K., *Magnetic Registers, Triggers and Gates*, Pt. 1, London, 1952
B.9 —, —, *ibid*, Pt. 2, London, 1953
B.10 BUCK, D. A., 'Ferroelectrics for Digital Information Storage and Switching', *M.I.T. Dig. Comp. Lab. Rep.* R–212, Cambridge, Mass., 1952
B.11 BURKS, A., GOLDSTINE, H. H., and VON NEUMANN, J., *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*, Princeton, 1946, 2nd ed., 1947
B.12 CHAMBERS, C. C. (Ed.), *Theory and Techniques for the Design of Electronic Digital Computers*, 4 Vols., Moore School, Philadelphia, 1947
B.13 CHEN, T. C., *General Principles of Crystal Diode Gating Circuits*, Moore School, Philadelphia, 1948
B.14 ECKERT, J. P., MAUCHLEY, J., and GOLDSTINE, H. H., *Description of the ENIAC*, Moore School, Philadelphia, 1945
B.15 E.R.A., *Summary of Characteristics of a Magnetic Drum Storage Computer*, St. Pauls, Minnesota, 1948
B.16 GILL, S., *Second Progress Report on the ACE*, London, 1949
B.17 GOLDSTINE, A. K., *Report on the ENIAC*, 2 Vols., Washington, 1946
B.18 GOLDSTINE, H. H., and VON NEUMANN, J., *Planning and Coding of Problems for an Electronic Computing Instrument*, Pt. 2, Vol. 1, Princeton, 1947
B.19 —, —, *ibid*, Pt. 2, Vol. 2, 1948
B.20 —, —, *ibid*, Pt. 2, Vol. 3, 1948
B.21 KITZ, N., *A Discussion of Calculating Machines*, M.Sc., Thesis, University of London, 1951
B.22 OLSEN, K. H., 'A Magnetic Matrix Switch', *M.I.T. Dig. Comp. Lab. Rep.* R–211, Cambridge, Mass., 1952
B.23 PATTERSON, G. W., *Proposed Programming of Matric-algebraic Operations on EDVAC*, Moore School, Philadelphia, 1947
B.24 —, *et al.*, *The EDVAC*, Moore School, Philadelphia, 1948
B.25 —, —, *Functional Description of the EDVAC*, 2 Vols., Moore School, Philadelphia, 1949

B.26 PETHERICK, E. J., 'A Small Sequence Controlled Calculator', *R.A.E. Tech. Note. ARM* 385, Pt. 1, 1947
B.27 SHARPLESS, T. K. (Ed.), *Progress Report on the EDVAC*, 2 Vols., Moore School, Philadelphia, 1946
B.28 SLUTZ, R. J., 'Memory Studies at the National Bureau of Standards', *U.S. Dept. of Commerce. Rep. No.* 2362, Washington, 1953
B.29 STIBITZ, G. R., *Function Theory of Counting Computers. Rep. No.* 1309, Burlington, Vermont, 1948
B.30 TOCHER, K. D., *Report on the Work of the Computer Group*, London, 1952
B.31 ULMAN, J. N., 'List of Laboratory Reports of Current Interest', *M.I.T. Dig. Comp. Lab. Rep.* R–173–2, Cambridge, Mass., 1951
B.32 WILKES, M. V. (Ed.), *Report of a Conference on High Speed Automatic Calculating Machines*, Cambridge, 1950
B.33 —, *Report on the Preparation of Programs for the EDSAC*, Cambridge, 1950
B.34 WILKINSON, J. H., *Progress Report on the ACE*, Pt. 1, London, 1947
B.35 —, *Report on the Pilot Model of the ACE*, Pt. 2, London, 1951

INDIVIDUAL COMPUTING MACHINES

ACE
Refs: B.16, B.34, B.35

C.1 (NO AUTHOR), *Electron. Engng*, 18 (1946) 372
C.2 WILKINSON, J. H., *Proc. roy. Soc.*, A.195 (1948) 265
C.3 WOODGER, M., *Nature, Lond.*, 167 (1951) 270

Analytical Engine. (*Babbage*)
Refs: A.6, A.7, A.37, A.48, A.11 p. 89, A.26 p. 69

C.4 BABBAGE, H. P., A.3 p. 13

APE(X)C
Ref: B.6

C.5 BOOTH, A. D., *Electron. Engng*, 22 (1950) 492
C.6 —, *ibid*, 24 (1952) 442
C.7 —, *Acta Phys. Austr.*, 4 (1950) 85
C.8 —, *J. Brit. Instn Radio Engrs*, 12 (1952) 587
C.9 —, and BRITTEN, K. H. V., *Quart. J. Mech.*, 2 (1949) 182

ARC
Ref: B.7

C.10 BOOTH, A. D., B.32 p. 17

ARRA

C.11 VAN WIJNGAARDEN, A., B.32 p. 113

ASCC (Harvard Mk.1.)
Ref: A.1

C.12 AIKEN, H. H., and HOPPER, G. M., *Elect. Engng, N.Y.*, 65 (1946) 384, 449, 552
C.13 BLOCH, R. M., A.3 p. 23
C.14 HARRISON, J. O., A.3 p. 208

BARK

C.15 KJELLBERG, G., B.32 p. 116
C.16 —, and NEOVIUS, G., *Math. Tab., Wash.*, 5 (1951) 29

Bell Laboratories Relay Calculators
C.17 ALT, F. L., *Math. Tab., Wash.*, 3 (1948) 1, 69
C.18 ANDREWS, E. G., *Elect. Engng, N.Y.*, 68 (1949) 751
C.19 —, A.4 p. 20
C.20 CESARES, O., *Bell Lab. Rec.*, 24 (1946), 457
C.21 JULEY, J., *ibid*, 25 (1947) 5
C.22 WILLIAMS, S. B., *ibid*, 25 (1947) 49
C.23 —, A.3 p. 40

BINAC
C.24 ECKERT, J. P., *et al.*, *Proc. Inst. Radio Engrs, N.Y.*, 40 (1952) 12

CALDIC
C.25 MORTON, P. L., *Math. Tab., Wash.*, 5 (1951) 57

Circle Computer
No published data

EDVAC
Refs: B.24, B.25, B.27

C.26 MAUCHLEY, J. W., A.3 p. 203
C.27 PATTERSON, G. W., A.4 p. 125

EDSAC
Refs: A.53, B.33

C.28 BENNET, J. M., *Distrib. Elect.*, 22 (1950) 251, 276
C.29 GILL, S., *Proc. Camb. phil. Soc.*, 47 (1951) 96
C.30 —, *Proc. roy. Soc.*, A.206 (1951) 538
C.31 WHEELER, D. J., *ibid*, A.202 (1950) 573
C.32 WILKES, M. V., *ibid*, A.195 (1948) 274
C.33 —, *Nature, Lond.*, 164 (1949) 341, 557
C.34 —, *J. sci. Instrum.*, 26 (1949) 217
C.35 WILKES, M. V., *Appl. sci. Res., Hague*, B.1 (1950) 429
C.36 —, *Nature, Lond.*, 166 (1950) 942
C.37 WILKES, M. V., and RENWICK, W., *Electron. Engng*, 20 (1948) 208
C.38 —, —, *J. sci. Instrum.*, 26 (1949) 385
C.39 —, —, *Math. Tab., Wash.*, 4 (1950) 61
C.40 —, —, B.32 p.9

ENIAC
Refs: B.14, B.17

C.41 BRAINERD, J. G., and SHARPLESS, T. K., *Elect. Engng, N.Y.*, 67 (1948) 163
C.42 BURKS, A. W., *Proc. Inst. Radio Engrs, N.Y.*, 35 (1947) 756
C.43 GOLDSTINE, H. H., and GOLDSTINE, A. K., *Math. Tab., Wash.*, 2 (1946) 97
C.44 HARTREE, D. R., *Nature, Lond.* 157 (1946) 527
C.45 —, *ibid*, 158 (1946) 500
C.46 MICHAEL, F. R., *Electronics*, 20 (1947) 116
C.47 TABOR, L. P., A.3 p. 31
C.48 WILKES, M. V., *Electron. Engng*, 19 (1947) 104

E.R.A. Computer
Ref: B.15

Harvard Mk. 2
Ref: A.2

C.49 CAMPBELL, R. V. D., A.3 p. 69

Harvard Mk. 3
C.50 MOORE, B. L., A.4 p. 11

Harvard Mk. 4
No published data

I.B.M. Computers
C.51 ECKERT, W. J., *Math. Tab., Wash.*, 3 (1948) 149
C.52 —, *Sci. Mon., N.Y.*, 67 (1948) 315

ICCE
Ref: B.30

Institut Blaise Pascal Computer
C.53 COUFFIGNAL, L., A.4 p. 374
C.54 —, B.32 p. 56

Institute for Advanced Study Computer
Refs: B.2, B.3, B.11, B.18, B.19, B.20
C.55 BOOTH, A. D., and BRITTEN, K. H. V., *Quart. J. Mech.*, 2 (1949) 182

LEO
C.56 PINKERTON, J. M. M., *Electron. Engng*, 23 (1951) 142

MADAM
C.57 KILBURN, T., *Nature, Lond.*, 164 (1949) 684
C.58 —, *ibid*, 168 (1951) 95
C.59 —, B.32 p. 119
C.60 THOMAS, G. E., B.32 p. 75
C.61 WILLIAMS, F. C., B.32 p. 26
C.62 —, and KILBURN, T., *Nature, Lond.*, 162 (1948) 487
C.63 —, *et al.*, *Proc. Instn elect. Engrs*, 98 (1951) 13
C.64 —, *Engineer*, 192 (1951) 47, 91, 126

MADDIDA
C.65 (NO AUTHOR), *Electronics*, 24 (1951) 244

MOSAIC
No published data

NAREC
C.66 GRINDLEY, D. H., and SARAHAN, B.L., *Electron. Engng*, 70 (1951) 111

OARAC
No published data

OMIBAC
C.67 LESTER, B. R., A.4 p. 65
C.68 (NO AUTHOR), *G. E. Rev.*, 53 (1950) 42
C.69 (NO AUTHOR), *Radio Electronics*, 21 (1950) 9

ORDVAC
No published data

RASCAL
No published data

Raytheon Computer
C.70 BLOCH, R. M., A.4 p. 50
C.71 BLOCH, R. M., *et al.*, *Math. Tab., Wash.*, 3 (1948) 286, 317
C.72 WEST, C. F., and DETURK, J. E., *Proc. Inst. Radio Engrs, N.Y.*, 36 (1948) 1452

SEAC
C.73 HARRISON, J. O., *Mat. Tab., Wash.*, 4 (1950) 164, 229
C.74 *Bull nat. Bur. Stand. Tech.*, 34 (1950) 121

SEC
Ref: B.21

C.75 BOOTH, A. D., *Proc. roy. Soc.*, A.195 (1948) 265
C.76 —, *Electron. Engng*, 21 (1949) 234

SWAC
C.77 HUSKEY, H. D., A.4 p. 83
C.78 —, *Mat. Tab., Wash.*, 4 (1950), 103
C.79 *Nat. App. Maths. Labs. Q. Rep.*, p. 1, 1950

TRE Computer
C.80 CARTER, R. H. A., B.32 p. 97
C.81 TUTCHINGS, A., B.32 p. 81
C.82 UTTLEY, A. M., B.32 p. 89

UNIVAC
C.83 SNYDER, F. E., and LIVINGSTON, H. M., *Mat. Tab., Wash.*, 3 (1949) 341
C.84 TUMBLESON, R. C., *Elect. Engng, N.Y.*, 67 (1948) 6

Whirlwind
Ref: B.31

C.85 FORRESTER, J. W., A.4 p. 44

Zuse Computer
C.86 LYNDON, C. R., *Mat. Tab., Wash.*, 2 (1947) 355
C.87 ZUSE, K., *Z. angew. Math. Mech.*, 30 (1950) 292

ARITHMETIC UNIT LOGIC AND DESIGN

Refs: A.5, A.22, A.26, B.2, B.3, B.5, B.6, B.11, B.12, B.14, B.17, B.21, B.24, B.25, B.27, B.30, B.34, B.35, C.6, C.42

D.1 BOOTH, A. D., *Quart. J. Mech.*, 4 (1951) 236
D.2 KATZ, J., *Proc. Inst. Radio Engrs*, 38 (1950) 202
D.3 RAE, T. J., and SPENCER, B. A., B.32 p. 41
D.4 ROBINSON, A. A., *Electron. Engng*, 25 (1953) 6
D.5 SHAW, R. F., *Rev. sci. Instrum.*, 21 (1950) 683

BIBLIOGRAPHY

COUNTERS AND BINARY ELEMENTS

Refs: A.5, A.22, A.32, A.36, A.41, B.2, B.3, B.12, B.31, D.2

Binary

E.1 ECCLES, W. H., and JORDAN, F. W., *Radio Rev., Lond.*, 1 (1919) 143
E.2 ISBORN, K., (Ferro-resonant), *Electronics*, 25 (1952) 121
E.3 REICH, H. J., *Electronics*, 12 (1939) 14
E.4 SCHMIDT, O. H., *J. sci. Instrum.*, 15 (1938) 24
E.5 WYNN-WILLIAMS, C. E., *Proc. roy. Soc.*, A.132 (1931) 295
E.6 —, *ibid*, A.136 (1932) 312

Ternary

E.7 BOOTH, A. D., and RINGROSE, J., *Electron. Engng*, 23 (1951) 133

Decimal

E.8 ALFVEN, H., *et al.*, (Trochotron), *Tekn. Tidskr., Stockh.*, 79 (1949) 169
E.9 BACON, R. C., and POLLARD, J. R. (Dekatron), *Electron. Engng*, 22 (1950) 173
E.10 BLISS, W. H., *Elect. Engng, N.Y.*, 68 (1949) 309
E.11 GROSDORF, I. E., *R.C.A. Rev.*, 7 (1946) 438
E.12 HOLLOWAY, D. L., *Electronics*, 25 (1952) 152
E.13 HOUGH, G. H., and RIDLER, D. S. (STC'tron), *Electron. Engng*, 24 (1952) 152, 230
E.14 NOBLE, S. W., B.32 p. 103
E.15 SHARPLESS, T. K., *Electronics*, 21 (1948) 122

FUNCTION TABLES (CODERS AND DE-CODERS)

Refs: A.5, A.22, B.10, B.22, C.42

F.1 BROWN, D. R., and ROCHESTER, N., *Proc. Inst. Radio Engrs, N.Y.*, 37 (1949) 139

GATES

Refs: A.5, A.22, A.26, A.30, B.2, B.3, B.12, B.13, B.14, B.15, B.21, B.25, B.27, B.31, C.5, C.8, C.37, C.42, C.56, C.63, C.72, C.76

G.1 BOOTH, A. D., *Electron. Forum*, 11 (1948) 5
G.2 CHEN, T. C., *Proc. Inst. Radio Engrs, N.Y.*, 38 (1950) 511
G.3 CLAYDON, D. O., *J. Brit. Instn Radio Engrs*, 13 (1953) 110
G.4 PAGE, C. H., *Electronics*, 21 (1948) 110, 118
G.5 ROSSI, B., *Nature, Lond.*, 125 (1930) 636
G.6 WILLIAMS, F. C., *J. Instn elect. Engrs*, 93 (1946) 289

INPUT-OUTPUT ORGANS

Refs: A.1, A.2, A.11, A.22, B.2, B.3, B.12, B.16, B.30

General

H.1 HOLLERITH, H. H., *J. R. statist. Soc.*, 57 (1894) 678
H.2 LENAERTS, E. H., *Electron. Engng*, 23 (1951) 140
H.3 RUBINOFF, M., A.3 p. 254
H.4 SHEPPARD, C. B., A.3 p. 267

Paper Tape

H.5 ALEXANDER, S. N., A.3 p. 248
H.6 MILLER, F. G., A.3 p. 213

Typewriters
H.7 WATSON, E. F., *Bell Syst. tech. J.*, 17 (1938) 620

Magnetic wire
Refs: B.2, B.3

H.8 WILLIS, D. W., B.32 p. 87

Cathode-ray Tube Output
H.9 BAKER, G. T., *J. Instn elect. Engrs*, 94 (1947) 421
H.10 FULLER, H. W., A.3 p. 238

Photographic
H.11 ELLIOTT, W. S., B.32 p. 85
H.12 GREIG, H. G., *Proc. Instn Radio Engrs, N.Y.*, 36 (1948) 1244
H.13 O'NEAL, R. D., A.3 p. 260

REGISTERS

Refs: A.22, B.2, B.3, B.8, B.9, B.21, C.5, C.72

Valve
I.1 KNAPTON, J. H., and STEVENS, L. D., *Electronics*, 22 (1949) 186

Core
I.2 KINCAID, M., *et al.*, *Electronics*, 24 (1951) 108
I.3 WANG, A., *Proc. Inst. Radio Engrs, N.Y.*, 38 (1950) 626
I.4 —, *ibid.*, 39 (1951) 401
I.5 —, and WOO, W. D., *J. appl. Phys.*, 21 (1950) 49
I.6 WOO, W. D., A.4 p. 91

RELAYS

Refs: A.2, A.11, A.22, A.30

J.1 BOWMAN, J. R., A.14 p. 119
J.2 BROOKER, R. A., *J. sci. Instrum.*, 29 (1952) 45
J.3 DAWSON, T., *Wireless World*, 59 (1953) 39
J.4 DEACON, E. L., *J. sci. Instrum.*, 25 (1948) 44
J.5 FRENCH, J. L., and LANE, G. H., *J. sci. Instrum.*, 26 (1949) 160
J.6 HUMPHRIES, H. E., *Engng Suppl. Siemens Mag.*, No. 110 (1934)
J.7 HYDE, N. E., *Electron. Engng*, 25 (1953) 11
J.8 JOEL, A. E., *Trans. Amer. Inst. elect. Engrs*, 67 (1948) 720
J.9 PIZER, H. I., *J. sci. Instrum.*, 25 (1948) 424
J.10 TAYLOR, D., and BENNETT, A. E., *J. sci. Instrum.*, 25 (1948) 283
J.11 SHANNON, C. E., *Trans. Amer. Inst. elect. Engrs*, 57 (1938) 713
J.12 —, *Bell Syst. tech. J.*, 28 (1949) 59

SINGLE DIGIT STORAGE

Refs: A.5, A.20, A.22, A.32, A.36, B.2, B.3, B.8, B.9, B.12, B.14, B.17, B.21, C.5, C.8, C.53, C.54, C.72, E.1, E.2, E.3, E.5, E.6, E.7, 1.2, 1.3, 1.4, 1.5, 1.6

STORAGE

Refs: A.11, A.20, A.22, B.4, B.10, B.12, B.15, B.21, B.22, B.24, B.25, B.27, B.28, B.31, B.34, B.35, C.5, C.7, C.25, C.37, C.50, C.53, C.54, C.59, C.60, C.61, C.63, C.72, C.81, C.85

Cathode Ray
Refs: A.22, B.31, C.59, C.61, C.63, C.85

L.1 BIVANS, E. W., and HARRINGTON, J. V., *Proc. Inst. Radio Engrs, N.Y.*, 38 (1950) 205
L.2 DODD, S. H., *et al.*, *Elect. Engng, N.Y.*, 69 (1950) 990
L.3 ECKERT, J. P., A.4 p. 32
L.4 —, *et al.*, *Proc. Inst. Radio Engrs, N.Y.*, 38 (1950) 498
L.5 FORRESTER, J. W., A.3 p. 125
L.6 HAEFF, A. V., *Electronics*, 20 (1947) 80
L.7 —, *Math. Tab., Wash.*, 3 (1948) 281
L.8 JULIAN, R. S., and SAMUEL, A. L., A.4 p. 96
L.9 WILLIAMS, F. C., *Proc. roy. Soc.*, A.195 (1948) 279
L.10 —, and KILBURN, T., *Proc. Instn elect. Engrs*, 96 (1949) 81, 183

Magnetic-acoustic

L.11 BOOTH, A. D., Prov. Pat. No. 16319/51
L.12 BRADBURD, E. M., *Elect. Commun.*, 28 (1951) 46
L.13 MILLERSHIP, R., *et al.*, *Brit. J. appl. Phys.*, 2 (1951) 304

Magnetic Core Matrix
Refs: B.22, B.31

L.14 FORRESTER, J. W., *J. appl. Phys.*, 22 (1951) 44
L.15 PAPIAN, W. N., *Proc. Inst. Radio Engrs, N.Y.*, 40 (1952) 475

Magnetic Drum
Refs: A.10, A.22, B.4, B.15, B.21, C.5, C.50, C.76, C.81

L.16 COHEN, A. A., *Math. Tab., Wash.*, 4 (1950) 31
L.17 KORNEI, O., A.3 p. 223
L.18 MOORE, B. L., A.3 p. 130
L.19 MORTON, P. L., *Proc. Inst. Radio Engrs, N.Y.*, 38 (1950) 211
L.20 POULSEN, V., *Ann. Phys., Lpz.*, 3 (1900) 754
L.21 SMITH, O., *Elect. World, N.Y.*, 1 (1888) 116
L.22 WILLIAMS, F. C., and WEST, J. C., *Proc. Instn elect. Engrs*, 98 (1951) 29

Photographic
Ref: L.18

L.23 TYLER, A. W., A.3 p. 146

Selectron
Ref: A.22

L.24 RAJCHMAN, J., *Math. Tab., Wash.*, 2 (1947) 359
L.25 —, A.3 p. 133
L.26 —, A.4 p. 365
L.27 —, *R.C.A. Rev.*, 12 (1951) 53

Sonic Delay Line
Refs: A.11, A.22, B.24, B.27, B.28, B.34, B.35, C.37, C.56, C.72, C.84

L.28 GOLD, T., *Phil. Mag.*, 42 (1951) 787
L.29 SHARPLESS, T. K., *Electronics*, 20 (1947) 134
L.30 —, A.3 p. 103

BIBLIOGRAPHY

COMPUTER APPLICATIONS

Refs: A.13, A.22, A.25, A.26, A.53, B.1, B.7, B.18, B.19, B.20, B.23, B.33

M.1 HAMMING, R. W. (Codes), *Bell Syst. tech. J.*, 26 (1950) 147
M.2 MCCALLUM, D. M., and SMITH, J. B. (Logic), *Electron. Engng*, 23 (1951) 126
M.3 MCCULLOCH, W. S., and PITTS, W. (Neural mechanisms), *Bull. math. Biophys.*, 5 (1943) 115
M.4 OETTINGER, A. G. (Learning), *Phil. Mag.*, 43 (1952) 1243
M.5 REDHEFFER, R. (Nim), *Amer. Math. Mon.*, 55 (1948) 343
M.6 SHANNON, C. E. (Chess), *Phil. Mag.*, 41 (1950) 256
M.7 ULAM, S. (Monte Carlo method), A.4 p. 207

NAME INDEX

Only those names mentioned in the text are included, other names will be found in the bibliography.

Page numbers set in bold type denote most important reference.

AIKEN, H. H. 10
BABBAGE, C. **7**, 12, 14, 15
BABBAGE, H. P. 10
BALDWIN, F. S. 6
BOOTH, A. D. 44, 203
BURKS, A. 17
CHEBYSHEV, P. L. **190**
CLEMENT, J. 7
COLEBROOK, F. M. 16
CREED, F. G. 71
DE COLMAR, C. X. T. 6
ECCLES, W. H. 13
ECKERT, J. P. 14, 17
FORRESTER, J. W. 18, 124, 132
GILL, S. 160, 169
GOLDSTINE, H. H. 17, 44, 151, 159
GOODWIN, E. T. 192
HAEFF, A. V. 132
HOLLERITH, H. 8
JORDAN, F. W. 13
KATZ, J. 42
LEGENDRE, A. M. 189, 191
LEIBNITZ, G. W. **5**
MAUCHLEY, J. W. 14, 17
MORLAND, S. 6
MÜLLER, J. H. 7
NEWTON, I. 181
OHDNER, W. T. 6
PASCAL, B. 6, 7
POULSEN, V. 13
REIFFLER, E. 206
RICHENS, J. H. 204
ROSSI, B. 82
ROTH, P. 6
SCHEUTZ, G. 8
SCHMIDT, O. H. 92
SMITH, O. 13
STANHOPE, C. 6
STIRLING, J. 181
TAYLOR, B. **189**
TURING, A. M. **16**
VON NEUMANN, J. **14**, 44, 144, 151, 159
WHEELER, D. J. 160, 169
WILKES, M. V. 16, 160, 169
WILLIAMS, F. C. 18, 130, 138
WOMERSLEY, J. 16

SUBJECT INDEX

Only those subjects mentioned in the text are included, others will be found in the bibliography.

Page numbers set in bold type denote most important reference.

Absolute transfer of control, 31, 33
Access time, 17, 23, 25, 113
Accuracy of computers, 2
A.C.E., 16, 17, 85, 150
Adder, 36, 38
  half, 38, 77
Analogue computer, 2
Analytical engine, 8, 9
'And' gate, 77
  order, 150
Anti-coincidence unit, 42
A.P.E.(X.)C., 19, 31, 146, 149, 153, 154, **157**, 170, 176, 180, 192, 193
A.R.C., 11, 12, 122
Arithmetic unit, 1, 22, 30, **35**
A.R.R.A., 12
A.S.C.C., 10
Atomic scattering factor, 201

Ballistic tables, 13
Barium titanate, 20, 126
B.A.R.K., 11
Bell laboratories computer, 10
Binary counting element, **93**
  -decimal conversion, 178, 179
  element, 30
  scale, 5, 35
Bi-quinary scale, 11
Birth, 214
Bistable flip-flop, 93
Boolean algebra, 77
Branching order, 4, 28
British Tabulating Machine Computer, 19

C.A.L.D.I.C., 20
Cash register, 2
Cathode coupled, anode output gate, 84, 85
  follower, 80
  output gate, 83
  ray tube display, 200
    store, 13, 18, 20, 103, **130**
Chain carry, 9
Character recognition, 61, 67
Chebyshev polynomials, **190**
Chess, 210, 211
'Circle' computer, 20
Clear, operations, 39
Clock pulses, 120
Code, 3, 17, 27, 136
Coding, 3
Co-incidence senser, 111, 112
  unit, 42
Collation, 58, 59
Complementary notation, 4, 35
Conditional transfer, 4, 9, 28, 34, 143, 208
Conditioned reflex, 211, 214
Contour plotting, **198**
Control combinations, 160
  counter, 25, 27, 29, 30
  of magnitudes, **167**
  organ, 1, 23, 25, 27, 32
  register, **25**
Core storage, single digit, 97
  type register, 107
Cosine function, 186, 191
Counter, binary, **93**
  ring, 52, 101

Decimal-arithmetic, 50
  -binary conversion, 175, 176
Decoder, **26**, 108
Dekatron, 100, 101
Delay line, 138
Deltamax, 87
Diagnostic aids, 72
Dictionary, storage of, **204**
Dielectric storage, 128
Difference engine, 7
Differential gate, 88
Digital computers, 2
Diode-gate, **78**
  -matrix function table, 110
  -network, 42
Divider, 36, 47
Division, 140, 144, 152, 155, 162, 163, 193, 195
Duty cycle, 22

E.D.S.A.C., 16, 160

Educating a machine, 214
E.D.V.A.C., 16
Electron density map, 197
Elusive '1', 44
Encouragement, 215
E.N.I.A.C., 14, 15, 173
E.R.A., '*1101*', 19

Ferranti computer, 18
Ferrites, 88, 89, 97, 126
Ferro-electric materials, 20, 134
  -resonant flip-flop, 98, 99
Film, 61, 66
Floating point arithmetic, **167**
Flip-flop, 28, 30, 92
Flow diagrams, **151**
Foreign language (*F.L.*), 203
Four-address code, 137
Function table, 14, 108
  many-one, 29, 108
  one-many, 29, 108

Games, 206
Gate, 26, 30, 31, **75**
  'and', 77
  diode, 78
  magnetic, 87
  multigrid, 86
  'not', 77
  'or', 77
  Rossi, 82
  series triode, 85
General purpose computer, 1
German, 204
Germanium crystal diodes, 21

Half-adder, 38, 77
  -subtractor, 43
Halving and doubling multiplier, 55
Harvard computer, Mk. 1, 10
  Mk. 2, 10
  Mk. 3, 19
  Mk. 4, 19
H.C.R., alloy, 87, 89
Hollerith tabulator, 8, **61**, 71

I.B.M., computers, 10, 11
I.C.C.E., 11
Immediate, instruction, 150
Income tax, 1
Initial orders, 160
Input/Output, 3, 34
Instincts, 215
Institute for Advanced Study computer, 17, 144, 146, 159, 160
Instruction, 23
Intelligence, 212, 216
Interpolation, 10, **180**

Iron nail store, 20
Iteration, 1, 3, 4, 143, 152, 155, 171, 195, 202
  index, 4, 152

Jacquard loom, 8, 9

Key switches, 61, 62, 73

Learning, 211
Least square error, 189, 190
Legendre polynomials, **189**
£ *s. d.*, 54
Linking orders, 162
Logical delay, single digit store, 96
Logic of drum store, 120, 121

M.A.D.A.M., 18
Magnetic, acoustic store, 97, 98, 123, 124
  disc store, 122
  drum store, 11, 13, **18**, 61, 115, 116, 134, 138, 148. 161, 214
  gates, 87
  tape, 61, 67, 72, 134
Magneto striction, 98
Mechanical translation (*M.T.*), 59, 69, 203
Memory, 23, 30, 113
Mercury delay line, 13, **15**, 103, 126, 128
Mersenne primes, 3
Mill, 9
Minimization process, 202
Mk. 1 computer, 10
Mk. 2 computer, 10
Mk. 3 computer, 19
Mk. 4 computer, 19
Monostable multivibrator, 92
Multi-grid, gate, 86
  valve, 86
Multiple induction loop, 153
Multiplication, 22, 28, 35, 44
  programme, 140
  time, 192
Multipliers, 36
  halving and doubling, 55
  matrix, 54
Multivibrator, monostable, 92
Musical output, 72

National Accounting Machine, 8
Neon lamps, 70, 73
Newton's interpolation formula, 181
Nickel wire, 98
'Not', gate, 77
  order, 150
Noughts and crosses, **207**
Numeroscope output, 61

One-address code, **136**
Optimum coding, 17, 138, 157, **192**
'Or', gate, 77
  order, 150
Order, 23
O.R.D.V.A.C., 19
Orthogonality, 190
Output, musical, 72
  vocal, 73

Parallel operation, 24, 30
Partial substitution, 57, 59
Pentaerythritol tetranitrate, **198**
Permalloy F, 87
Phase angle, 198, 199
Photoelectric tape reader, 62
Pin store, 114
Pluggable sequence relay calculator, 11
Polynomial expansions, **189**
Post-editor, 206
Powers calculator, 8
Pre-editor, 206
Princeton register, 107
Printing time, 193
Programming, 3, 136
Punched, cards, 161
  tape, 161
Punishment, 214

Random numbers, 216
Raster, 66, 131
Read around constant, 131, 132
Read/record head, 115, 119
Reading, 23
Rectifier, 78, 81
Refinement, 201, 202
Register, 36
  core, 107
  Princeton, 105, 106
  shifting, **103**, 139
  Wang, 107
Relay, 28, 75, 90
  tree, 109
Remanence, ferromagnetic, 19, 20
Resistance matrix, 109, 110
Ring counter, 52, 101
Rossi gate, 82
Rounding off, 2

Saturable reactor function table, 110, 111
Scale of two, 5
Schmidt trigger, 92, 93
S.E.A.C., 17
S.E.C., 19
Secondary emission, 19, 129
  coefficient, 129
Selectron, 17, 18, 133, 134
Selenium diode, 81
  matrix, 93
Semantic vacuity, 205
Serial operation, 24, 30
Series triode gate, 85
Shift instructions, 37, 146, 159
Shifting register, **103**, 139
Shopping, 212, 213
Short tank, 103
Simple induction loop, 152
Sine function, **182**
Single digit store, 90
Size of store, 134
Slide rule, 2
Solenoid, 71
Sonata, 211
Special purpose computer, 1
Spoken word input/output, 61
Square root, 34, 143, 144, **169**
S. T. C.-tron, 100, 101
Sterling, 54
Stirling's formula, 181, 182
Storage capacity, 9, 15, 134
Store, 9, 23, 113
Subroutines, 16, 149, 158
Subtraction, 22, 28, 43
Subtractor, 36, 43
  half, 43
Supersonic flight, 3
S.W.A.C., 19
Symbolic logic, 77
Syntax, 204

Tables, storage of, 65
  use of, 146, 159, 180, 181
Tape, punched, 161
Target language (*T.L.*), 203
Taylor series, **189**
Teletype, 12, 61
  printer, 61, 62, 70, **200**
  tape, 16, 18, 61, 62
Terminal organs, 3, 26, 28, 61
Tetrode, 86
Three-address code, 137
Three state storage device, 89
Thyratron, 91
Track and word location, 194
Transfer order, 138
Transistors, 21
T.R.E. computer, 18
Tree, relay, 109
Trigger tube, 102
Triode, **81**
Trochotron, 100, 101
Two-address code, 17, 19, 136, 154, 196

Unit cell, 197, 201
U.N.I.V.A.C., 17

Use of additional drums on *A.P.E.* (*X.*)*C.*, 149

Vocal output, 73
Vocoder, 73
Voice-operated input, 69

Wang type register, 107
Wetting of relay contacts, 77
Whirlwind, 18
Whist, 206, 207
Wire store, 114
Writing, 23, 31

X-ray crystallography, 187, 197
  diffraction spectrum, 197

Züse computer, 11

S10.78
B

DATE DUE	DATE DUE
10/27/6/	
JL 19'63	

WESTINGHOUSE ELECTRIC CORP.
AIR ARM DIV. - LIBRARY

WESTINGHOUSE ELECTRIC CORP.
AIR ARM DIV. - LIBRARY

Acc. 2087
Booth & Booth
copy 2
510.78
B

AUTOMATIC DIGITAL CALCULATORS

BOOTH & HUSKEY

510.78
B